Vehicular Traffic Flow Prediction Model Using Machine Learning-Based Model

Jiahao Wang

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

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. It is always one of the hot topics about how to implement an efficient, robust and accurate vehicular traffic prediction system.

With the help of Machine Learning-based (ML) methods, especially Deep Learning-based (DL) methods, the accuracy of the prediction model is increased. However, we also noticed that there are still many open challenges under ML-based vehicular traffic prediction model real-world implementation. Firstly, the time consumption for DL model training is relatively huge compared to parametric models, such as ARIMA, SARIMA, etc. Second, it is still a hot topic for the road traffic prediction that how to capture the special relationship between road detectors, which is affected by the geographic correlation, as well as the time change. The last but not the least, it is important for us to implement the prediction system in the real world; meanwhile, we should find a way to make use of the advanced technology applied in ITS to improve the prediction system itself.

In our work, we focus on improving the features of the prediction model, which

can be helpful for implementing the model in the real word. Firstly, we introduced an optimization strategy for ML-based models' training process, in order to reduce the time cost in this process. Secondly, We provide a new hybrid deep learning model by using GCN and the deep aggregation structure (i.e., the sequence to sequence structure) of the GRU. Meanwhile, in order to solve the real-world prediction problem, i.e., the online prediction task, we provide a new online prediction strategy by using refinement learning. In order to further improve the model's accuracy and efficiency when applied to ITS, we provide a parallel training strategy by using the benefits of the vehicular cloud structure.

The following papers are part of this work under the supervision of Prof. A. Boukerche:

- J. Wang and A. Boukerche.: The Scalability Analysis of Machine Learning-Based Models in Road Traffic Flow Prediction, IEEE ICC'20 - NGNI Symposium
- A. Boukerche and J. Wang: Vehicular Traffic Flow Prediction System Using Hybrid Machine Learning-Based Model, Elsevier - Ad-Hoc Networks
- A. Boukerche and J. Wang: Machine Learning-Based Traffic Prediction Models for Intelligent Transportation Systems, Elsevier - Computer Network
- J. Wang and A. Boukerche: Non-parametric models with optimized training strategy for vehicles traffic flow prediction, Elsevier - Computer Network

Acknowledgements

First of all, I would like to express my sincerest appreciation to my supervisor, Prof. Azzedine Boukerche, for his guidance, support, patience, and encouragement for my entire path of graduate study. His enthusiasm for knowledge has always inspired me considerably and invigorated me to move forward. Without his understanding and precious help, I would not have been able to complete my research. I am so honored and proud of working with such a responsible and excellent supervisor. I am very grateful for all the vast amount of benefits I received from being one of his students—not only the knowledge itself but also the positive influence of his attitude towards life.

Furthermore, I thank my parents, who have always given their unconditional support and love to me. Throughout my research, they have been my reliable harbor when I have encountered challenges. I feel so lucky to be their son and receive their love.

In addition, I appreciate the experience of working in the PARADISE Lab and the valuable advice from Dr. Sun.

Last but not least, I would like to thank all the members of PARADISE Lab for their help, cooperation, and understanding and for being great friends.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As noted in [1], there will be almost 7 billion people living in urban areas in the year 2050, comprising two-thirds of the world's population. With the progress of urbanization and the popularity of automobiles, transportation problems are becoming more and more challenging: vehicles are congested on the road, traffic accidents are frequent, and the traffic environment is deteriorating. How to improve the capacity of the road network has attracted more and more scholars' attention. The first solution that occurs to most

of us is to build more highways and expand the number of lanes on the road to solve this problem. However, according to the study done by Dechenaux et al. [2], the expanding road capacity will cause more severe traffic conditions.

Many great ideas proposed easing the transportation system's burden and improving the traffic environment, where the smart city is one of the most famous ideas for future urbanism [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 5, 17] . A smart city integrates the urban resources by using big data technology, Artificial Intelligence (AI), cloud computing, the Internet of Things (IoT), communication technologies, etc [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51] . The smart city system can help the governor of the city regulate the resources more efficiently and reduce pollution. Meanwhile, the smart city system can also provide the citizen a more comfortable and more intelligent living environment. The Intelligent Transportation System (ITS) is a vital part of the smart city system [52, 53, 54, 55, 56] . ITS is a complex system integrated by a variety of advanced technologies, e.g., transportation communication systems. Meanwhile, by taking the advantages of the development of the communication system [57, 58], abundant on-road sensors [59, 60, 61, 62], etc., ITS can provide real-time road infrastructure analysis, more efficient road traffic control [63], and fast vehicular cloud (VC) service. With the help of ITS, a safe, mobile, environmentally sustainable, and comfortable road environment can be provided [64].

To achieve ITS, an accurate and efficient vehicular traffic flow prediction system is needed [65, 66] . The role of the vehicular traffic flow prediction system to ITS is to provide punctual continuous and precise road status information based on road conditions (such as vehicular traffic flow trends and volume). That information can be helpful for the applications involved in ITS, such as traffic congestion control, traffic light control, vehicular cloud (VC), etc [67]. Here, we take VC as an example. One difficulty in implementing and maintaining a VC is to computing the available redundant vehicular

resources on a given road segment so as to better determine the possible workload of the cloud. However, the on-road sources are mainly gathered from the high mobility vehicles on the highway or an urban road, making it so important for the cloud system to determine how many vehicles there will be on the given road segment in the future. 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 gives the VC opportunity to simulate the future workload and possible computing capacity [68, 69, 70, 71, 72, 73] .

The difficulty with implementing a vehicular traffic flow prediction system into ITS can be divided into the following three parts: accuracy, efficiency, and online prediction.

The origin of the accuracy problem for the prediction system is the feature of the vehicular traffic patterns. The vehicular traffic patterns, as time-series, are non-stationary, and their variations are affected by vehicular traffic lights, changes in weather, and other factors. Some of these play a decisive long-term role, making the variation show specific trends and certain regularities. Others play a short-term role, introducing some uncertainty into the variation. The variation of time-series can be divided into the following types: (1) Trend variation, which means that processes change with time and show a trend of continuous rise, decline, or steadiness in a certain direction; (2) periodic change (seasonal change), which let the processes show cyclical fluctuations at a fixed period, it also refers to the unfixed periodic fluctuation of the processes; (3) random variation, which means the processes are affected by accidental factors and exhibit irregular variations [74]. Time-series 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 bring great difficulties in accuracy in predicting vehicular traffic flow.

As for the prediction model's efficiency, it can be seen as the combination of cost of implementation and prediction. The implementation cost is mainly caused by the train-

ing phase of the prediction model and implementing this model into a large road network. It takes much time to train a prediction model, especially for a Machine Learning-based (ML) model with a Deep-Learning structure [75, 76] . Meanwhile, when implementing the model to a large road network, the different pattern features on different road segments may require models with different parameters on each road segment, which is a massive cost for training the prediction model for each road segment.

The last difficulty is to achieve online prediction. As one of the core components of ITS, VC brings edge computing into the transportation system by utilizing the spare computing and storage capacity on-road with the help of Ad hoc network [77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100] . This brings faster computation and communication speed in the VC, but it has a smaller storage space and relatively weaker computation capacity than traditional center cloud computing. Meanwhile, it is not cost-effective to update the vehicular traffic flow information whenever new data arrives. The above features lead to the consequence that vehicular traffic flow does not update in real-time. A realizable online prediction strategy should be proposed for the vehicular traffic flow prediction system to overcome this problem.

1.2 Objectives

According to the difficulty we addressed before, the objectives of our thesis are as follows:

- Evaluate the performance of several famous state-of-the-art non-parametric models and their Deep Learning structures on both freeway dataset and urban dataset. We focus on the model's accuracy and training cost and the efficient optimization algorithm, reducing the implementation cost of a prediction model with a complex inner structure.

- Propose a new traffic flow prediction system using Machine Learning-based model

that has better prediction accuracy than the other state-of-art models. Meanwhile, the model should have high efficiency. Last but not least, the model should have good features in dealing with the extraction of spatial-temporal features for a big road network.

- Design and implement a system scheme that can help apply the prediction system to a big road network. Meanwhile, it is better to utilize the benefits from the VC.

- Design and implement a strategy for online prediction, dealing with the data delay and data loss.

1.3 Contributions

The main contributions of this paper are summarized as follows.

- We provide an off-line optimization algorithm called Desensitization, and its improved version–Combined Training Strategy (CTS), reduces the time cost in the training process. According to the experiment results, we can reduce lots of training time by using Desensitization while implementing the models to the whole road network. The training time for implementation is reduced to less than 10% compared to the training time while using the normal method. While using CTS, the training time is longer than using Desensitization. However, the training time using CTS is still half shorter than using the regular training strategy. Meanwhile, the CTS process assures the pre-trained model's qualification on basic feature extraction by embedding the Desensitization process. The refinement process of CTS can further improve the model accuracy. Furthermore, the three verification stage reduce the model's overfitting risk. CTS improves the model efficiency without compromising on accuracy.

- We propose a new performance metric called Gain, which is used to evaluate the

cost-effectiveness of the model.

- A hybrid prediction model is proposed. A Graph Convolutional Network (GCN) using max diffusion is used at the beginning of the model. The following Attention-based Sequence to Sequence model takes the outputs from the GCN as input to further extract the spatial-temporal features in the original dataset. In the experiment, we showed that the diffusion GCN is more effective and efficient than multi-layer GCN. Meanwhile, by comparing our model with other models (S2S, SA2S, DARNN, etc.), we showed that the spatial-temporal feature extraction phase is a good way to improve the accuracy of the prediction model. Our model also showed better robustness while used for online prediction tasks.

- An efficient training strategy is proposed, which can improve accuracy and implementation efficiency for a large-scale road network. According to the experience results, the strategy can maintain the smellier accuracy rate than the traditional training strategy by cost 1/5 training time on each grid.

- An online prediction process based on refinement learning has also been proposed. With the help of the new prediction process, the model can maintain relatively high accuracy when facing different lengths of time lag. According to the experiment result, our proposed method can significantly improve prediction performance by only costing a relatively small extra training time.

1.4 outline of the Thesis

This paper is organized as follows.

- Chapter 2 reviews the background and related works of Machine Learning-based models.

- Chapter 3 proposes the optimization algorithm, traffic flow prediction model, and the strategies we designed.
- Chapter 4 provides the experiment results of the model and strategies we present in Chapter 3.
- Chapter 5 concludes the advantages of the whole prediction system and discusses planned future work.

Chapter 2

Background and Related Work

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 section, 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.

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

$$\mathfrak{A}(\mathrm{X}, \mathcal{F}, \mathrm{G}) \rightarrow \widehat{\mathrm{X}}, \tag{2.1}$$

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. $\widehat{\mathrm{X}} \in \mathbb{R}^{C_r * M_r * N_r}$ 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 $\widehat{\mathrm{X}}$: Traffic flow prediction, traffic speed prediction,

etc. For example, in [101], 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 [102], 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 $\widehat{\mathrm{X}}$:
  - If $C_r > 1$: Multi-task prediction task. For example, in [103], 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 [104], 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 $\widehat{\mathrm{X}}$: Short-term prediction task, long-term prediction. For example, in [105], the prediction intervals were from 30-min to 120-min, which can be considered a long-term prediction. While in [106], the prediction interval was set to be 5-min, which is a short-term prediction task.

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 2.1 according to [107, 108, 109, 110], 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 sam-

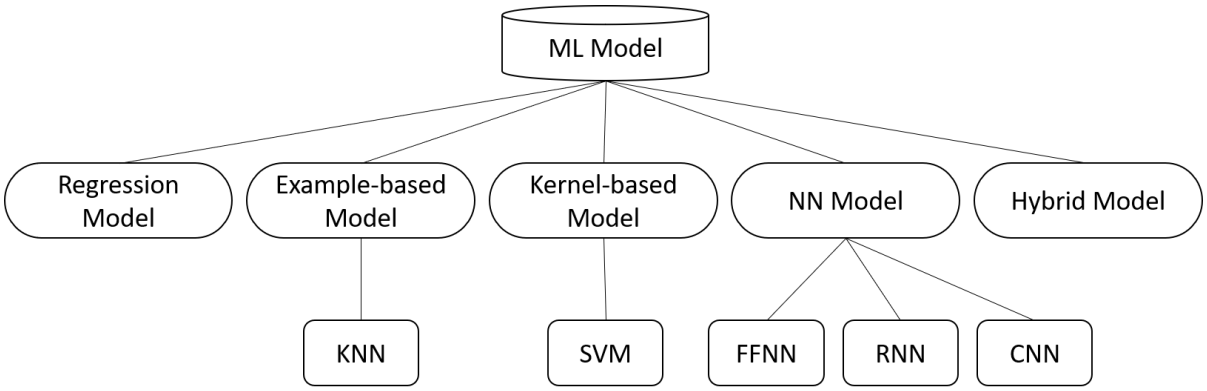


Figure 2.1: ML-based models

ples, 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).

- 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.

This Chapter is organized as follows: In Section 2.1, 2.2 and 2.3, we will review the existing ML models applied in traffic prediction based on the previously mentioned

taxonomy. In Section 2.4, we will review the open-source datasets that can be used for the traffic prediction task. In Section 2.5, 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 2.6.

2.1 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.

2.1.1 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 [111], 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 [112, 113], 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 [114] 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 [115]. 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. [116] 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 [117] 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 [101] 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 2.1. 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

Table 2.1: Regression models used for traffic prediction in the past ten years

Papers	Tasks	Regression Model	Input-Output	Area	Additions
Okawa et al. (2017)	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. (2015)	Flow	LASSO λ=50000	One-One	Freeway	Based on Granger causality theory; Two Lasso phase.

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 [114, 115] 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 [101], 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.

2.1.2 Example-based model

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 [118], 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.

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 [119]: (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 [120], 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 [121] 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 [122], the authors 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

Table 2.2: KNN models used for traffic prediction in the past ten years

Papers	Task	State vector	Distance Metric	Prediction Method	Additions
Chang et al. (2012)	Flow	Target detector's historical records	Euclidean distance	Weighted average	Grid search for k and d
Zhang et al. (2012)	Flow	Target detector's historical records	Euclidean distance	Weighted and non-weighted average	k=18, d=4; Data preprocess use Mix-Max Normalization and reasonable check
Zheng and Su (2014)	Flow	Target detector's historical records	Correlation coefficient distance	LSPC	k=10
Xia et al. (2016)	Flow	Target road & upstream & downstream historical records	Distance consider spatial-temporaltraffic	Weighted average and trend adjustments	K=4; MapReduced KNN model;
Cai et al. (2016)	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.

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 [102] chose another strategy. In [122], 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 [102] 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 2.2 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 [122]. 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 [120, 123]. 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 [102], 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.

2.1.3 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 [124]. 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.

Table 2.3: 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,...}

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 2.3.

SVR was first found to be used in the traffic prediction filed in [125]. 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 [126], 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 minutes, 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 [127] tried to solve the optimization problem of SVR in a better way.

The new SVR with additional parameters in [127] 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 [128, 129] both used RBF as their kernel function. Meanwhile, some scientists have tried to consider seasonal features in the kernel function. In [130], 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 [129] 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 [131]; 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 [132]. 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 [133], 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 [134], 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

Table 2.4: The SVM model proposed in the past ten years

Papers	Kernel	Parameters	I/O	Area	Additions
Castro-Neto et al. (2009)	RBF		One-One	Urban	Online prediction
Hong et al. (2011)	RBF	Optimized by CACO	One-One	Urban	
Wang and Shi (2013)	Mexican-hat wavelet function	a=6, α=0.0001, C=1000	One-One	Freeway	Embed input data by G-P approach.
Lippi et al. (2013)	RBF + Linear (Season)	S=480 S-RBF: C=5, ϵ=0.01, γ=1, γ^S=192, ω=3 S-Lin: λ=0.001, ϵ=0.01, ω=3, T=150000	One-One	Freeway	
Mingheng et al. (2013)	RBF	Optimized by Grid search	One-One	Urban	Online prediction
Duo et al. (2017)	RBF, Polynomial, Linear for different frequency fluctuation	Optimized by GPSO C=98.24, ε=0.021, σ=0.68	One-One	Urban	decompose the original sequence by EMD
Luo et al. (2019)	RBF		One-One	Freeway	Make decomposition using DTF

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 [128], 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 [135], 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 2.4, 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 accuracy. In [128, 135], 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 [135]. However, methods such as CACO [129] and GPSO [132] are used to quickly find the hyper-parameters' values.

2.2 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 [119]. 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 [136, 119] 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 [137]. 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 [138]. 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

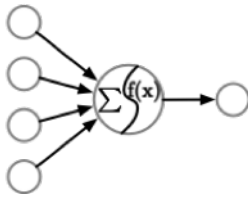


Figure 2.2: Typical structure of a neuron in neural network

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.

2.2.1 Feed Forward Neural Network (FFNN)

Artificial neurons are the basic units of neural networks; a typical structure of a neuron is shown in Fig. 2.2. If we assume that a neuron takes d inputs $x_1, x_2, ..., x_d$, and the input vector to the neuron can be defined as $(x) = [x_1, x_2, ..., x_d]$. The output of the neuron will be:

$$a = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.2}$$

where $\mathbf{w} = [w_1, w_2, ..., w_d]$ 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

Table 2.5: 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)

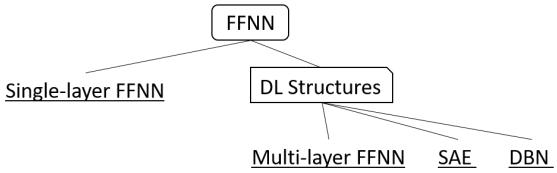


Figure 2.3: Taxonomy of the analyzed state-of-the-art literature related to FFNN

listed in Table 2.5.

Each neuron in the FFNN is divided into different groups, according to the order of receiving information. Each group can be be seen as 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 [108].

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 a detailed taxonomy of the analyzed state-of-the-art literature related to FFNN is shown in Fig 2.3.

The simplest and most user-friendly structure for FFNN is one-hidden layer structure. In [139], 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 [140]. 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 [141], 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.

2.2.1.1 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 [142], 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 [143] trained the Multi-layer FFNN in an unsupervised way by using the method called Wasserstein Generative Adversarial Network (WGAN) [144]. 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 Belif 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. 2.4. 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 [145].

According to [145], 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 $X = [x_1, x_2, ..., x_N]$, and the output of hidden layer is Y, in AE we have:

$$\begin{aligned} Y &= f(W_e X + b_e), \\ Z &= f(W_d Y + b_d), \end{aligned} \tag{2.3}$$

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

$$L(X, Z) = \frac{1}{2} \sum_{i=1}^{N} ||x_i - z_i||^2, \tag{2.4}$$

The AE model used in [145] 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

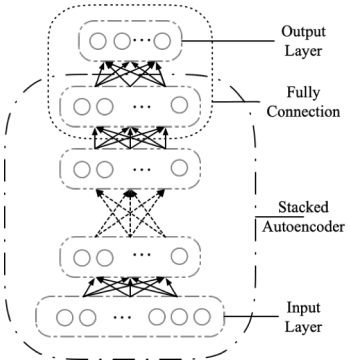


Figure 2.4: Structure of stacked autoencoders

$\widehat{rho_j}$ over the training set for the j-th neuron in the hidden layer as follows:

$$\widehat{\rho_j} = \frac{1}{N_T} \sum_{i=1}^{N_T} y_j(X^{(i)}), \tag{2.5}$$

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 calculated as follows:

$$KL(\rho||\widehat{\rho_j}) = \rho log\frac{\rho}{\widehat{\rho_j}} + (1-\rho)log\frac{1-\rho}{1-\widehat{\rho_j}}, \tag{2.6}$$

KL divergence has the feature when ρ is equal to $\widehat{\rho_j}$, $KL(\rho||\widehat{\rho_j}) = 0$. Then we can rewrite the cost function 2.4 to:

$$L(X,Z) = \frac{1}{2}\sum_{i=1}^{N} ||x_i - z_i||^2 + \gamma \sum_{j=1}^{H_D} KL(\rho||\widehat{\rho_j}), \tag{2.7}$$

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 [145]. These models were tested by making 15, 30, 45, and 60 minute 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 [146, 147]. 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 $\mathbf{v} = [v_1, ..., v_m]^T$ and hidden units as $\mathbf{h} = [h_1, ..., h_n]^T$, the energy function of RBM can be defined as:

$$\begin{aligned} E(v, h) &= -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i,j} v_i w_{ij} h_j, \\ &= -a^T v - b^T h - v^T W h, \end{aligned} \tag{2.8}$$

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:

$$\begin{aligned} p(v_i = 1|h) &= \sigma(a_i + \sum_j w_{ij} h_j), \\ p(h_j = 1|v) &= \sigma(b_j + \sum_i w_{ij} v_i), \end{aligned} \tag{2.9}$$

The free energy of RBM can be calculated as:

$$\mathcal{F}(v) = -a^T - \sum_j log(1 + e^{(b_j + \sum_i w_{ij} v)}), \tag{2.10}$$

and the weights and bias can be updated by calculating:

$$\begin{aligned} -\frac{\partial \log p(v)}{\partial w_{ij}} &= E_v[p(h_i|v)] - v_i \cdot \sigma(W_i \cdot v), \\ -\frac{\partial \log p(v)}{\partial b_j} &= E_v[p(h_j|v)] - \sigma(W_i \cdot v), \\ -\frac{\partial \log p(v)}{\partial a_i} &= E_v[p(h_i|h)] - v_i, \end{aligned} \tag{2.11}$$

The units are updated by using Markov chain as follows:

$$\begin{aligned} h^{(n+1)} &\sim \sigma(W^T \cdot v^{(n)} + b), \\ v^{(n+1)} &\sim \sigma(W \cdot h^{(n+1)} + a), \end{aligned} \tag{2.12}$$

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 [103]. 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 [103]. In [148], the authors also used a DBN for traffic flow prediction; the difference is that the authors adopted FireFly Algorithm (FFA) [149] 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 [150], 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

Table 2.6: FFNN models used for traffic prediction in the past 10 years

Papers	Task	I/O	NN structure	Optimization	Additions
Jin and Sun (2008)	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. (2010)	Flow	One-One	One hidden layer FFNN		
Kumar et al. (2013)	Flow	One-One	One hidden layer FFNN (6-neurones)	Levenberg-Marquardt	Sensitive analysis
Lv et al. (2015)	Flow	One-One	SAE		
Polson and Sokolov (2017)	Speed	Multi-One	Multi-layer FFNN	SGD	l_1 trend filter for data preprocessing
Lin et al. (2018)	Flow	One-One	Five FFNN layer (activator: Leaky ReLU ; 128 hidden units)	Adam	Detrending for data preprocess; WGAN for unsupervised training
Goudarzi et al. (2018)	Flow	One-One	Three layer DBN	Gradient descent	FFA used for hyper-parameters optimization
Zhao et al. (2019)	Flow	One-One	Two layer DBN (100 units each)		Parallel pre-training and fine-tune

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 2.6. 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 [143]. However, as we can see from Table 2.6, 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.

2.2.2 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

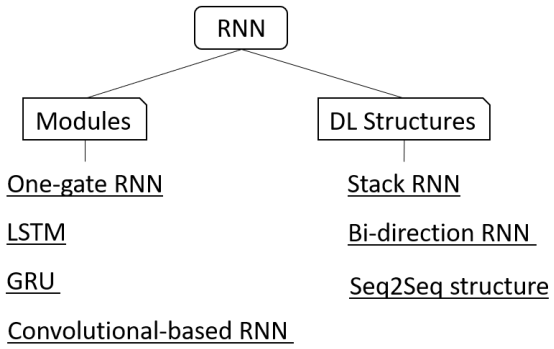


Figure 2.5: Taxonomy of the analyzed state-of-the-art literature related to RNN

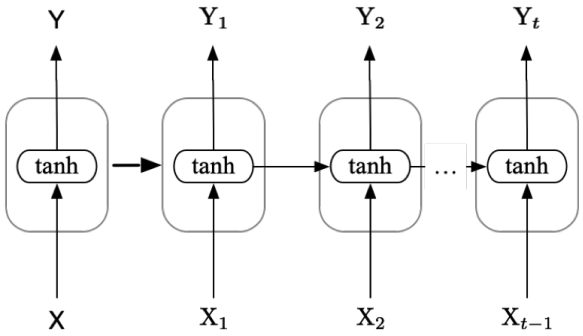


Figure 2.6: Structure of a basic RNN and unfolded by time

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 [151].

The idea of RNN was proposed in the 1980s by M.I. Jordan et al. [152, 153, 154]. 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 a detailed taxonomy of the analyzed state-of-the-art literature related to RNN is shown in Fig 2.5.

An RNN can be unfolded into several repeat modules based on time, as we can see in Fig 2.6. Inside a standard RNN repeat module, there is only one gate. If we assume the input time sequence is $X = [x_1, x_2, ..., x_{t-1}]$, where x_i is the data observed at time i, And the output vector is $H = [h_1, h_2, ..., h_{t-1}]$, and h_i is the output based on the i-th

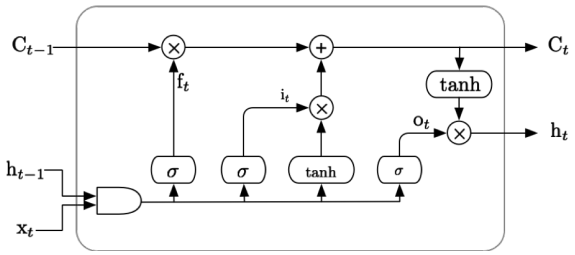


Figure 2.7: Structure of the repeating module in LSTM

input and the output in previous timestamp, which can be shown as:

$$h_i = F(W \cdot [h_{i-1}, x_i] + b), \tag{2.13}$$

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 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 [155]. 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. [156] 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. [157], 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. 2.7. 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.

$$\begin{aligned} f_j &= \sigma(W_f \cdot [h_{j-1}, x_j] + b_f), \\ i_j &= \sigma(W_i \cdot [h_{j-1}, x_j] + b_i), \\ o_j &= \sigma(W_o \cdot [h_{j-1}, x_j] + b_o), \end{aligned} \tag{2.14}$$

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:

$$\begin{aligned} \widetilde{C_j} &= tanh(W_c \cdot [h_{j-1}, x_j] + b_c), \\ C_j &= f_j * C_{j-1} + i_j * \widetilde{C_j}, \end{aligned} \tag{2.15}$$

And the output of the module at time t is:

$$h_j = o_j * tanh(C_j), \tag{2.16}$$

As we can see from the above functions and Fig. 2.7, the LSTM module takes the current time input data and the hidden state from the previous timestamp as its input. In [158, 159], 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:

$$
\begin{aligned}
f_j &= \sigma(W_f \cdot [C_{j-1}, h_{j-1}, x_j] + b_f), \\
i_j &= \sigma(W_i \cdot [C_{j-1}, h_{j-1}, x_j] + b_i), \\
o_j &= \sigma(W_o \cdot [C_j, h_{j-1}, x_j] + b_o).
\end{aligned}
\tag{2.17}
$$

In [160], 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 [161]. The structure of the GRU module is shown in Fig. 2.8. As we can see, there are

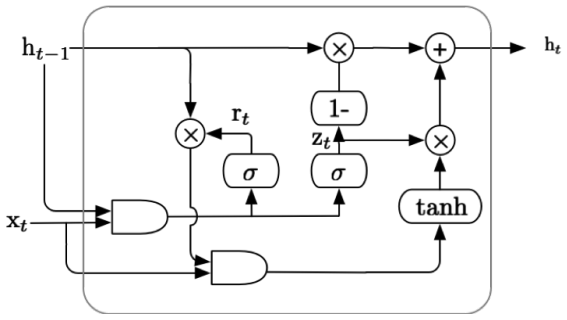


Figure 2.8: Structure of the repeating module in GRU

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:

$$\begin{aligned} r_i &= \sigma(W_r \cdot [h_{i-1}, x_i] + b_r), \\ z_i &= \sigma(W_z \cdot [h_{i-1}, x_i] + b_z), \end{aligned} \tag{2.18}$$

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

$$\begin{aligned} \widetilde{h_i} &= tanh(W_h \cdot [r_i * h_{i-1}, x_i] + b_h), \\ h_i &= (1 - z_i) * h_{i-1} + z_t * \widetilde{h_i}. \end{aligned} \tag{2.19}$$

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 [104]. 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 [162] 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 minutes by using data from the previous 30 minutes. 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 [163]. 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:

$$
\begin{aligned}
f_j &= \sigma(W_{fc} \odot C_{j-1} + W_{fh} \cdot h_{j-1} + W_{fx} \cdot x_j + b_f), \\
i_j &= \sigma(W_{ic} \odot C_{j-1} + W_{ih} \cdot h_{j-1} + W_{ix} \cdot x_j + b_i), \\
\widetilde{C_j} &= tanh(W_c \cdot [h_{j-1} + x_j] + b_c), \\
C_j &= f_j \odot C_{j-1} + i_j \odot \widetilde{C_j}, \\
o_j &= \sigma(W_{oc} \odot C_j + W_{oh} \cdot h_{j-1} + W_{ox} \cdot x_j + b_o), \\
h_j &= o_j \odot tanh(C_j),
\end{aligned}
\tag{2.20}
$$

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.

2.2.2.1 Deep-Learning structure for RNN

To stimulate the RNN's greater potential, Pascanu et al. [164] and Hihi et al. [165] brought RNN into a Deep-Learning structure. As was pointed out in [166], applying deep structure to the RNN module can highly improve the accuracy of the prediction model. A deep structure 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. 2.9. 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. 2.9, L_i^j is the RNN module in layers j at timestamp i, N is the number of layer in SRNN. If

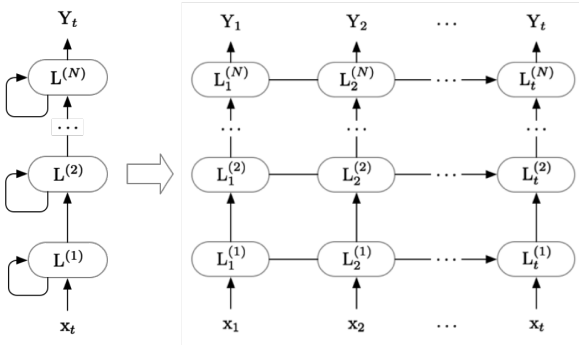


Figure 2.9: SRNN structure and SRNN unfolded based on time series

we assume $h_i^{(j)}$ as the output of the j layer at time i, $h_i^{(j)}$ is calculated as follows:

$$h_i^{(j)} = f(U^{(j)} h_{(i-1)}^{j} + W^{(j)} h_i^{(j-1)} + b^{(j)}), \tag{2.21}$$

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 [167]. 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 hour. 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 overfitting situation. To avoid this, the authors in [168] 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

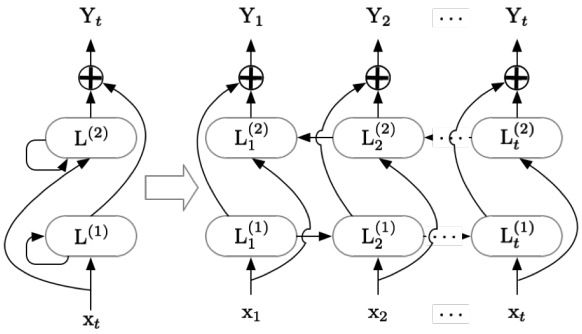


Figure 2.10: Bi-RNN structure and Bi-RNN unfolded based on time series

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 overfitting 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 [169] called Bidirectional Recurrent Neural Network (BRNN), as shown in Fig. 2.10. 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:

$$\begin{aligned} h_i^{(j_1)} &= f(U^{(1)} h_{(i-1)}^1 + W^{(1)} x_i + b^{(1)}), \\ h_i^{(j_2)} &= f(U^{(1)} h_{(i+1)}^1 + W^{(2)} x_i + b^{(2)}), \\ Y_i &= Concatenate(h_i^{(j_1)}, h_i^{(j_2)}). \end{aligned} \tag{2.22}$$

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 [170], 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 [171], 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 [172], 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

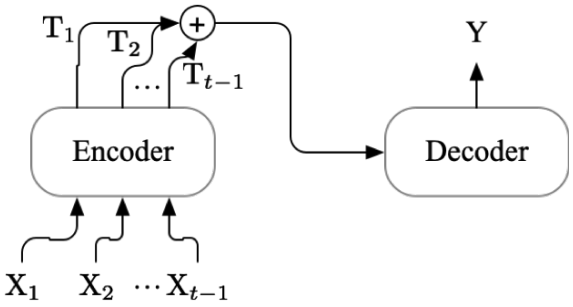


Figure 2.11: Structure of Seq2seq structure

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 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 [173, 161]. The structure of seq2seq is shown in Fig. 2.11. 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 $C = [h_1, ..., h_t]$ by using the hidden states h from the RNN modules inside the Encoder in each timestamp, which can

be calculated as follows:

$$h_i = R(x_i, h_{i-1}), \tag{2.23}$$

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 [174, 105].

The application of seq2seq can be seen in [105]. 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 minutes 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 [174] 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 data set.

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 2.7. 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-

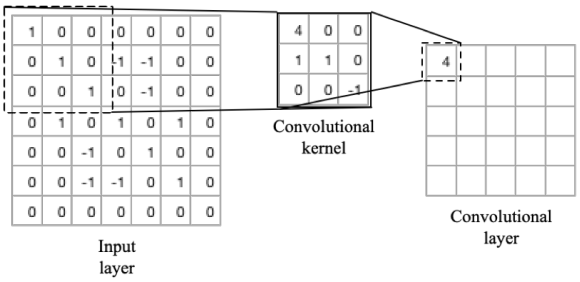


Figure 2.12: An example of a simple convolution process for which the size of the input layer is 7*7, the size of convolutional kernel is 3*3, the size of moving step is 1, and the size of convolutional layer is 5*5

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.

2.2.3 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 [175]. 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

Table 2.7: RNN based models in the past ten years

Type	**Papers**	**Task**	**Prediction Interval**	**Area**	I/O	Structure	Additions
LSTM	Luo et al. (2019)	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. Classfiy the historical data by different workdays.
	Yi et al. (2019)	Speed	5-min	Freeway	One-One	SRNN (LSTM)	GRU outperforms LSTM while the input length is shorter
	Yang et al. (2019)	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. (2018)	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. (2018)	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. (2018)	Flow	1-hour	Urban	One-One	SRNN (LSTM)	
	Tian et al. (2018)	Flow	5, 15, 60-min	Freeway	One-One	SRNN (LSTM-M) (6 layers)	Using extra parameters to deal with data lost
	Wang et al. (2017)	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. (2017)	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. (2017)	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 (2015)	Flow	15, 30, 45, 60-min	Freeway	One-One	Single layer LSTM	
GRU	Zhao et al. (2019)	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 (2017)	Flow	5 30 -min	Freeway & Urban	One-One	Single layer GRU	Online multi-step model
ConvRNN	Yu et al. (2017)	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. (2019)	Speed	5, 15, 30, 60-min	Urban	Multi-Multi	Seq2seq (Conv-GRU) with attention layers in encoder and decoder	

high-order features. A simple convolution process is shown in Figure 2.12. 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:

$$C_{p,q} = \sum_{i,j} W_{i,j} * X_{i+p*l-1,j+q*l-1} + b \tag{2.24}$$

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. [176] provided a Graph-Convolutional NN (GCN) by introducing the Laplacian matrix into Convolutional NN. In [177], 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 [178], 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 [179], 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 2.8. 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 [179].

2.3 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

Table 2.8: CNN-based models used for traffic prediction in the past 10 years

Papers	Task	I/O	structure	Additions
Yu et al. (2017)	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. (2019)	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. (2019)	Speed	Mutli-Multi	3D-GCN with GLU activate function	Using Temporal adjacency matrix; Train the model based on both MAE and MSE

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 [180].

We will first take a look at the first type. In [181], 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 [182] 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 [183], 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 [184], 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

Table 2.9: Hybrid models used for traffic prediction in the past 10 years.

Papers	Task	Sub-Models	Combination Method	Additions
Tan et al. (2009)	Flow	Moving average, exponential smoothing, ARIMA	NN	Separated model used for historical records collected in different periods.
Moretti et al. (2015)	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. (2018)	Flow	CNN, GRU	Regression layer	Use attention layer in the beginning to extract the spatial-temporal correlation for the sub-models
Gu et al. (2019)	Flow	GRU, ARIMA, RBFNN	Improved Bayesian Combination method	Correlation analysis for choosing records for prediction.

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 2.9 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 [181] are used for predicting traffic flows containing different amounts of random vibrations; while in [183], 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 [182]. 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.

Table 2.10: The overview comparison of different ML model type.

Model	Regression	Example-based	Kernel-based	NN			Hybrid
				FFNN	RNN	CNN	
Implementation difficulty	Low	Middle	Middle	Middle	High	High	High
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
Temporal feature extraction	No	No	No	No	Yes	No	Yes
Spatial 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

2.3.1 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 2.10, we compare the models in the following aspects:

- Difficulty of implementation: How difficulty to implement the model, including figure out the structure of the model, the parameters or the hyper-parameters of the model. The accuracy of the Regression, example-based and Kernel-based models heavily depends on the parameters. It will cost much effort to find an adequate set of parameters. However, by using a proposed Machine-Learning kit, such as sci-learn kit [185], we can reduce the implementation difficulty. As for the NN-based model, depending on the complex network structure, we have to define the hyper-parameters for the model. There are also correlations among the hyper-parameters, such as the learning rate and the number of neurons in each layer. It means we will need more effort to find a perfect hyper-parameter set.

- Cost of implementation: It takes the training time and hardware dependency of the model into consideration. Usually, 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 model's performance? 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., the KNN model, also has a relatively high requirement on the dataset to build up a reliable candidates search pool.

- Deep-learning structure: Does this type of model has a Deep-Learning structure. All models under the NN category can be aggregated into a DL structure. For FFNN, we have multi-layer FFNN. There are multi-layer structures, bi-direction structures, sequence-to-sequence structures for RNN. As for CNN, we have multi-layer CNN combining CNN with several pooling layers. For the hybrid model, the DL structure is based on the sub-models we chose for the complete model.

- Ability of spatial feature and temporal feature extraction: As for this, we mean whether the model has a clear structure to extract a specific feature. For example-based models, the spatial features can be represented by combing historical data from interrelated road sections. For the CNN model, especially the Graph Convolutional Network (GCN), the spatial features can be represented by and learned from the adjacency matrix generated based on geographic information of road detectors. Meanwhile, for the temporal feature extraction, the RNN model extracts information from the time series using its recurrent structure.

- Time cost of Prediction: How much time will the model cost make the final prediction. Usually, it is highly related to the structure of the model. The more complex the structure is, the longer time it will cost. The prediction time cost of the example-based model depends on the size of the candidate database. While for the NN-based model, it depends on the number of parameters of the network.

Meanwhile, the recurrent structure of RNN will also increase the model calculation time.

- Maintain cost: It includes the update difficulty of the model hyper-parameters; for KNN, it also includes the updates of the candidate search pool. As for NN-based mode, usually, it will require a re-training process to update the parameters.

2.4 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 data set 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.

2.4.1 Datasets collected from real-word

First, we will focus on the datasets collected from the real-word on-road sensors [186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198]. 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 [199]: 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 minutes, 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 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 [200]: 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 minutes. 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 [201]: The traffic flow and occupancy information were collected every 30 seconds and aggregated into 5 minutes. 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 [202].

2.4.2 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 [203].

In macroscopic traffic simulation, traffic flow is considered continuous compressible fluid, composed of many vehicles [204]. 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 [205], EMME [206].

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 [207]. 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 [208], VISSIM [209], SimTraffic [210], CORSIM [211], Paramics [212], MITSimLab [213], TransModeler [214].

2.5 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.

2.5.1 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 [215], 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 [142], 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 [134], the authors chose the Discrete Fourier Transform (DFT) as the detrending method. In [216, 132], 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 [217], Probabilistic Principal Component Analysis (PPCA) is used to deal with the missing data.

2.5.2 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 [103, 218]. In [104], 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 [172, 219]. As pointed out in [148], the original CNN cannot deal with the graph information. To use the graphed spatial features (adjacency matrix), [178, 177] 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 [178, 174, 105]. 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 [220, 221]. In [179], 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 [167, 160]. 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 [103], FFA [149], ABC [222], etc.

2.6 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.

2.6.1 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.

2.6.2 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.

Chapter 3

Methodology

In this chapter, we will introduce the methods used to build up the prediction system. Inside of just proposing a prediction model, we also provide the strategies which can be helpful for implementing the model into ITS.

3.1 Data preprocessing

3.1.1 Moving Average

As pointed out in [134, 223], vehicular traffic flow records that have too short record interval will contain too many unstable vibrations, which is not only useless for making driving or control decisions, but also affects the accuracy of the prediction model. However, an interval that is too long will lose many seasonal features for vehicular traffic data. In order to transform a short interval record into the desired interval duration, the Moving Average (MA) method is used in this paper. The MA for the first n records can be calculated as follows:

$$X'_1 = \frac{1}{n}\sum_{i=n}^{1} X_i, \tag{3.1}$$

where X_i represent the record at time i. To calculate the following records, we use

$$X'_m = X'_{m-1} + \frac{1}{n}(x_m - x_{m-n}). \tag{3.2}$$

3.1.1.1 Data normalization

The role for the normalization method is to map the original data records into a given range. The reason to do so is that most of the activation functions used in NN model, such as Sigmoid function or Tanh function, map the input into a fixed range. However, the range for the original records varies indeterminately. Thus, we use the normalization function to make the original records fit for the activation functions, and can help the model converge faster. In our work, we use MinMaxScaler function to normalize the vehicular traffic flow data, which can be represented as follows:

$$\begin{aligned} X_{std} &= \frac{(X - X_{min})}{(X_{max} - X_{min})}, \\ X_{scaled} &= X_{std} \times (X_{max} - X_{min}) + X_{min}, \end{aligned} \tag{3.3}$$

where X is the original data and X_{std} is the data after scaling. This standardized method is to scale the data between 0 and 1.

3.2 Training optimization algorithm

We need to consider the time cost implementing a prediction system using ML-based models, especially when it is in a DL structure. Usually, a model trained for a given dataset can only be used for the specific dataset; the model's accuracy will be highly decreased if we force the model to predict other datasets. It means the model for all detectors on a given transportation network may share the same structure. The time cost will be huge if the system needs to create a new model for each detector and then train each model on a relative dataset one at a time to ensure the whole prediction system's accuracy.

To reduce the overall training time while implementing an ML-based model, we provide an off-line optimization algorithm called desensitization in [109], as shown in Alg. 1. The algorithm tries to avoid redundant training process during the implementation. As we can see in [109], even though different detectors' records collected at the same time of the day may be different, the overall traffic flow trend is almost the same in one-day differences can be further minimized by using normalization algorithms. This feature allows us to reuse a well-trained model's parameters over the whole network with appropriately fine-tuning the parameters.

However, the desensitization algorithm still has its disadvantages. As we can see in 1, to ensure the model's accuracy for one specific dataset, the model is trained inside a while loop to meet the threshold requirement. However, focusing on the training accuracy will easily cause an overfitting problem, especially when the training dataset's size is small. This situation will be more general when the model's structure is relatively complicated. To further improve the desensitization algorithm's performance, we proved

Algorithm 1 Desensitization algorithm

Input: : $mList$: list of untrained models, $dList$: datasets for untrained models, th: value of threshold, ne: number of retrain epochs;
step 0: Train base model using historical average dataset -> trained parameters $tPara$.
for *model in mList and ds in dList* **do**
  step I: copy $tPara$ into *model*;
  step II: Test *model* on $ds.test -> PE$ // Predict error;
  if $PE > th$ **then**
    step III.a Update the parameter in *model* using Back propagation in ne epochs;
    step III.b Go to **step II**;
  end
end
Return $mList$

a new optimization algorithm called Combined Training Strategy (CTS).

As we can see in Fig. 3.1, CTS combines the Desensitization algorithm with the idea of model refinement. For each detector, The pre-trained model is first verified by using both its training dataset (X_{train}) and verification dataset (X_{verify}), and we can get R-square values R^b_{train} and R^b_{verify} for the base model, respectively. Then, the process will first inspect whether R^b_{train} is higher or equal to the training gate value G_{train}. This process is to check whether the pre-trained model fits the basic requirement extracting the overall dataset trend. If not, the pre-training will be sent to the Desensitization process; otherwise, the model goes directly into the next stage. After the model finishing the Desensitization, CTS will test the model using X_{verify} to get R^d_{verify}. Then, CTS compares R^d_{verify} with R^b_{verify}, and the model with higher R-square value goes to the next step. The CTS test whether the model's R_{verify} meets the verification gate requirement G_{verify}. If yes, the model can be used for future prediction task; otherwise, the model will go to the refinement stage.

As shown in Fig. 3.2, the basic idea of model refinement is to train an assistant model with a simple structure, using the reconstructed training data. There are two main stages inside the model refinement process. Firstly, the base model is only used for reconstructing the training data. We remove the record on the first timestamp and append the prediction result on $n + 1$-th timestamp at the end of the input sequence.

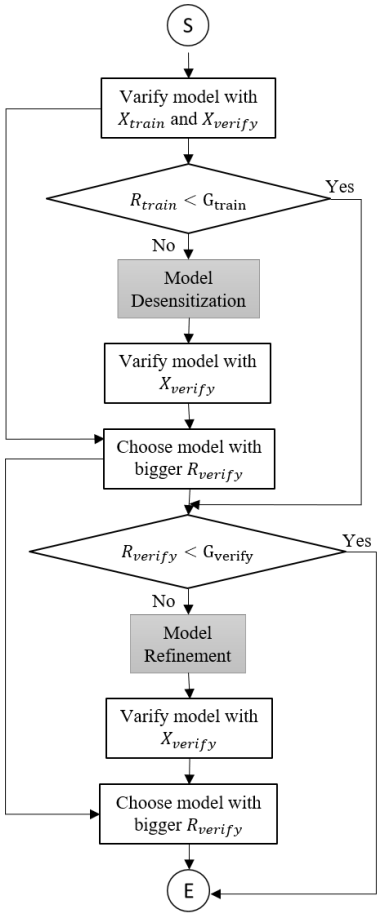


Figure 3.1: Overview on the CTS process.

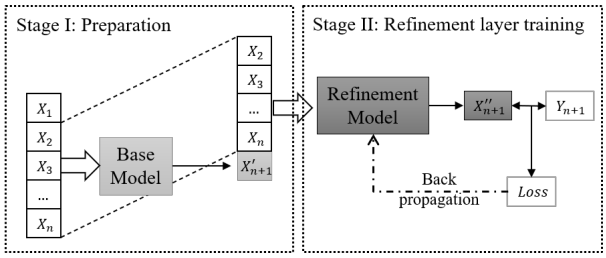


Figure 3.2: Overview on the model refinement process.

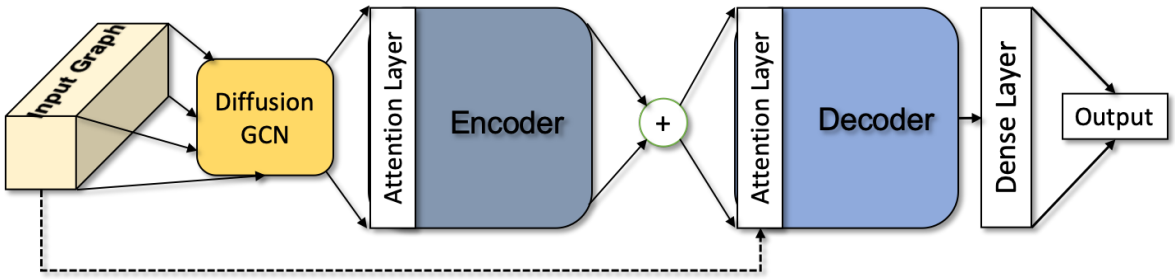


Figure 3.3: Overview of the basic structure of the prediction model

This new sequence is used for training the refinement layer. Secondly, we tune the parameters inside the refinement layer using Back Propagation based on the model's loss. The prediction will require both the base model and the refinement layer, while the training process in model refinement is limited to the refinement layer. By doing this, we can further improve the model's accuracy without spending too much time. Moreover, by treating the refinement layer as an independent part of the original base model, we can reduce overfitting risk while performing additional training loops.

After model refinement, we get R^r_{verify} by verifying the model on X_{verify}. Then, we compare R^r_{verify} with R_{verify}, the model with higher R-Square value will be used for the final prediction. As we can see, CTS keeps an eye on the model's verification value whenever the model finishes a training process. It will help us to avoid the overfitting caused in each training process.

3.3 Model design

In this section, we will take a look at the prediction model. As we can see from Fig 3.3, the model can be divided into two main parts, i.e., the GCN layer, and the Attention-based sequence to sequence structure using bi-directional GRU kernel. The first layer is used to extract the spatial relationship among different data collectors based on their geographic information. Then, in the following part, the spatial-temporal feature is further extracted based on the vehicular traffic flow records.

3.3.1 GCN

As we have said at the beginning of the section, the task for the first part of the model is to extract spatial correlations in the road network. As we know, the vehicular flow in a given road section is affected by the road traffic changes on its neighbors and other, more distant road section; thus, closer correlations in the road section result in greater vehicular traffic change affection transported to the given road section. The traditional method is to use a weighted matrix to represent the connection relationship between detectors. However, this method is challenging to apply to large and complicated road network. As we know, CNN is widely used for extracting space relation for a given picture. However, one drawback of using traditional CNN in road traffic prediction tasks is that the geographic relation information we use is represented by adjacency matrix. However, the traditional CNN can only fit for the grid-structure picture, and has difficulty processing non-Euclidean structure data. To overcome this problem, Defferrard et al. provided a Graph-Convolutional NN (GCN) in [176].

The core of traditional CNN is the convolution compute, which can be defined as following:

$$(f * g)(t) = \int_{\mathbb{R}} f(x)g(t-x)dx, \tag{3.4}$$

where g is the filter on f. According to the definition of Fourier transform, inverse Fourier transform, and Convolution theorem, we could get:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}, \tag{3.5}$$

where $\mathcal{F}$ and $mathcalF^{-}1$ are Fourier transform and inverse Fourier transform.

The base for traditional Fourier transform is $e^{-2\pi ixv}$, while for the Fourier transform a graph, the base is the n dimensional eigenvector $U^{-1} = [u_1, \cdot, u_n]^{-1} \in \mathbb{R}^{n*n}$, where n is the number of nodes in the graph. So that, we can rewrite the Fourier transform on

graph as:

$$\mathcal{GF}\{f\}(\lambda_l) = \sum_{i=1}^{n} f(i)u_l^*(i), \tag{3.6}$$

where $f(i)$ is the signal on the i-th point, and u_l is the l-th line in U. If we define $x = (f(1), \ldots, f(n)) \in \mathbb{R}^n$, we can rewrite Equation 3.6 as follows:

$$\mathcal{GF}\{x\} = U^T x. \tag{3.7}$$

Meanwhile, we can rewrite the inverse Fourier transform as follows:

$$\mathcal{IGF}\{x\} = Ux. \tag{3.8}$$

By using 3.7 and 3.8 we can rewrite 3.5 as follows:

$$\begin{aligned} g * x &= \mathcal{IGF}\{\mathcal{GF}\{g\} \cdot \mathcal{GF}\{g\}\}, \\ &= U(U^T g \cdot U^T x), \end{aligned} \tag{3.9}$$

where g is the filter in Graph Convolution compute. To give the filter function the ability to aggregate the information from the adjacency nodes of the node in graph, the Laplacian matrix L is introduced to the function as $g(L)$, where L can be represented as $L = U\Lambda U^T$ and Λ is the diagonal matrix built up by the characteristic vector. $U^T g(L)$ can be further written as $g_\theta(\lambda)$. Then, Equation 3.9 can be rewritten as follows:

$$g_\theta * x = U g_\theta \cdot U^T x, \tag{3.10}$$

which is Equation (3) in [224]. According to the simplification process in [224], in the end we get the convolution function used for graph:

$$g_\theta * x = \theta(D^{-\frac{1}{2}}\widetilde{A}D^{-\frac{1}{2}})x, \tag{3.11}$$

Table 3.1: Commonly used Laplacian matrix representations

Type of L	**Representation**
Combinatorial Laplacian	$L = D - A$
Symmetric normalized Laplacian	$L^{sys} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
Random walk normalized Laplacian	$L^{rm} = D^{-1} L$

where $\widetilde{A} = I + A$, and A is the adjacency matrix of the graph, D is the degree matrix, which can be calculated as follows:

$$D = \begin{cases} \sum_{t=1}^{n} A_{it} & i = j, \\ 0 & otherwise, \end{cases} \tag{3.12}$$

The most commonly used Laplacian matrix representations are shown in Table 3.1, and in [224], the authors chose symmetric normalized Laplacian. In the end, by adding the activation function to equation 3.11, we get the final GCN function for layer l:

$$H^l = \sigma(D^{-\frac{1}{2}} \widetilde{A} D^{-\frac{1}{2}} H^{l-1} W^{l-1}) \tag{3.13}$$

In [225], the author provided the Diffusion CNN [226] on graph; by using the random walk normalized Laplacian, the diffusion convolution compute can be represented as follows:

$$x * g_\theta = \sum_{k=0}^{K-1} (\theta_{k,1} (D^{-1} A)^k + \theta_{k,2} (D A^T)) x, \tag{3.14}$$

where k is the number of truncation steps of the diffusion process

3.3.2 Attention-based sequence to sequence neural networks

With the help from GCN, the influence from geographical correlation can be perfectly captured. However, as pointed out in [179], the spatial correlation between each road section is changed with time. To face this challenge, an Attention-based sequence to sequence neural network is proposed, as shown in Fig. 3.4. The neural network can be

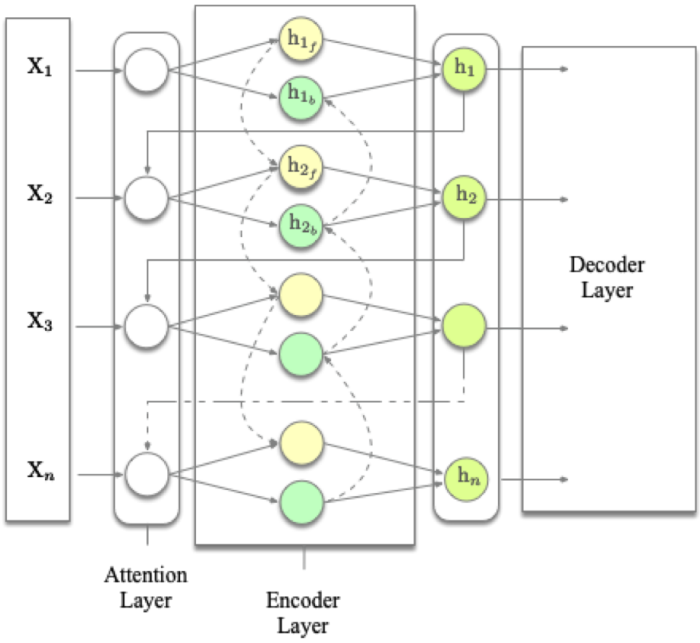


Figure 3.4: Legend of Attention-based sequence to sequence neural network

divided into two parts, i.e., the encoder with Attention layer, and the decoder. The data processed by the diffusion GCN is input into the Attention-based encoder, and the Attention layer in this part is used to extract the spatial correlation based on the time-series itself. The output from encoder will be inputted to the decoder. The final prediction will be made by a fully connected layer using the output from the decoder.

According to the research in recent years, the sequence to sequence model performs well in time-series prediction tasks[174, 105]. As we can see from Fig. 3.4, 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 $C_e = [h_{1_e}, ..., h_{t_e}]$ by using the hidden states h from the RNN modules inside the Encoder in each timestamp, which can be calculated as follows:

$$h_{i_e} = R_e(x_i, h_{i-1_e}), \tag{3.15}$$

where R_e can represent a single layer RNN or RNN in Deep-Learning structure inside the encoder. In our work, we choose GRU in a bi-directional structure to get a relatively faster computing speed; meanwhile, the bi-directional structure will let each hidden state

contain information not only from the previous timestamp, but also the timestamp behind it. The vector C is believed to contain the high-level features in the input sequence. Then, the RNN modules inside the Decoder will not only take the previous timestamp's output, but also the vector C to calculate its output as follows:

$$h_{i_d} = R_d(h_{1_e}, h_{i-1_d}). \tag{3.16}$$

The purpose of using the Attention layer is to determine how much information will be taken from each element in the input, by using normalized exponential function, which is shown as follows:

$$atten(X) = Softmax(\sigma(WX)), \tag{3.17}$$

where w is the weight matrix for the Attention layer. Adding Attention layer to encoder and decoder, we can rewrite Equation 3.15 and 3.16 as follows:

$$\begin{aligned} h_{i_e} &= R_e(atten_e(x_i, h_{i-1_e}), h_{i-1_e}), \\ h_{i_d} &= R_d(atten_d(h_{1_e}, h_{i-1_d}), h_{i-1_d}). \end{aligned} \tag{3.18}$$

3.4 Efficient training strategy

Traditional road traffic prediction systems, usually work for a large road network, and are built in a central computing center where all historical data is stored and updated; it is also where the prediction model is trained. This kind of system requires a huge computing capacity, and there will be a high frequency and large amount of network occupation for data updates. To overcome these problems, we co-opted ideas from parallel computing and distributed computing [227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240]. We provide a efficient training strategy, which will highly reduce the dependency on hardware. Meanwhile, the strategy takes the benefits from vehicular cloud (VC) and makes faster and more accurate predictions.

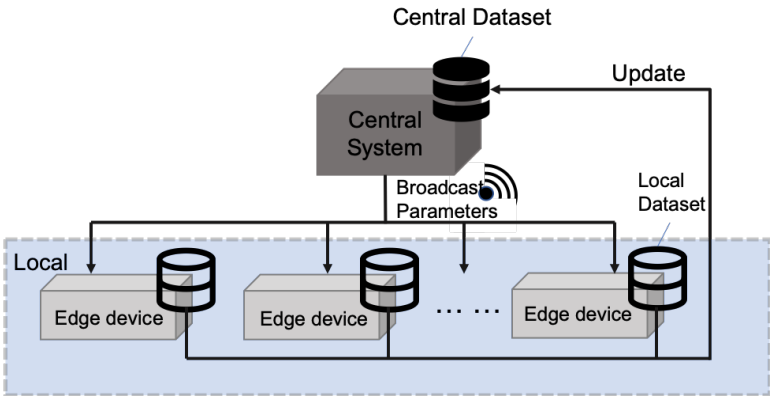


Figure 3.5: Legend of efficient training strategy

The overview on the strategy is shown in Fig. 3.5. As pointed out in [241, 242], a VC is a type of an edge cloud which is built by the road side units (RSU) and the vehicle on the road. The on-wheel resources and roadside infrastructures are aggregated by using cellular communications, Wireless Access in Vehicular Environment (WAVE), Dedicated Short Range Communication (DSRC), etc. Meanwhile, a centralized Cloud is set for providing necessary service. This kind of cloud structure is suitable for our training strategy.

In our design, the large road network will be separated into small grid, and each grid will have the same number of prediction points. A base model will be trained in the central cloud by using the historical data from all data detectors. The difference compared to tradition prediction system is that the size of the base model is the same with each small grid. After pre-training, the parameters of the base model will be broadcast to the local cloud in each grid. The prediction model in each local cloud has the same structure as the base model in the centre cloud. Then, each local cloud will fine-tune the parameters of the model with the data collected in their own grid. The central dataset is updated by taking the data stored in each local cloud with each preset updated time interval, such as every week or longer. By doing this, we can first reduce the network load between local and centre cloud. Second, we can achieve a flexible data update strategy. Last but not the least, the local model will be updated based on the

latest data, while the base model can keep good seasonal features by training with older but longer data collections.

3.5 Online prediction strategy

For road traffic prediction, the online prediction is an important topic to face. This is due to data transport and update delay. Meanwhile, the online prediction strategy is also useful for data loss.

To achieve online prediction, we use the continuous prediction method, where current prediction results are used for data completion and prediction for the next timestamp, which can be represented as follows:

$$\widetilde{Y}_{t+n} = P(Y_{t+n-i}, .., \widetilde{Y}_t, P(Y_{t-i+1}, .., \widetilde{Y}_{t+1}), .., \widetilde{Y}_{t+n-1}), \tag{3.19}$$

where i is the length of the time-series we use for prediction, and n is the number of time intervals by which the data is delayed.

3.5.1 Refinement learning

Furthermore, to get a more accurate prediction result, we provide the idea of a refinement learning. The aim of refinement learning is to improve the non-stationary of the model, without change the main structure of the trained model. Refinement learning gives the trained model to adapt the time interval length shift.

A refinement layer is added to the end of the original model, which is already well trained. The mechanism of the refinement layer is to learn the difference between the predicted data and the real data. As we can see from Fig. 3.6, the refinement layer takes the prediction results from the trained model as its input. The refinement layer is trained to learn the distance between the prediction from the give trained model to the real data records. The predicted distance and the real distance are used to calculate the loss value

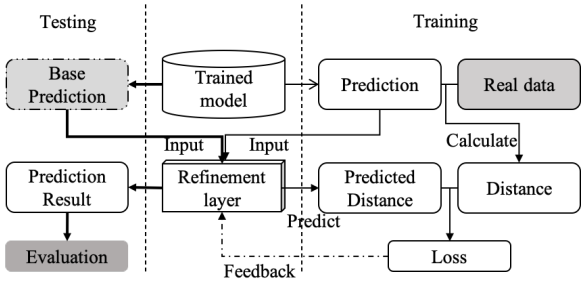


Figure 3.6: An overview of refinement learning mechanism.

as follows:

$$Loss = \frac{1}{2}(f_{rf}(\widetilde{Y}) - (\widetilde{Y} - Y))^2, \tag{3.20}$$

where f_{rf} represents the ML model calculation inside the refinement layer. The loss is used for back-propagating the refinement layer. The training process for the refinement layer is isolated from the main model, and will not change the parameters inside the main model. This will avoid us from spending lots of time re-training the based model, whose model structure is more complicated than the refinement layer. After the refinement layer is well trained, the trained model and refinement layer work together to make the final prediction for the test cases.

Chapter 4

Experiments and Evaluations

In this chapter, we will present the details of the implementation of our work. There will be two main parts of the experiments.

The first part will give a comprehensive analysis of the performance of the ML-based models and the deep-learning structures used for **One-One** traffic flow prediction task. The performance of a model includes the accuracy of the model and the efficiency of the model. Moreover, we will further discuss the cost of using deep-learning structures in traffic prediction tasks. In the end, we will give a short analysis of the Desensitization algorithm's performance.

The second part will test the new hybrid model we have proposed used for **Multi-Multi** traffic prediction tasks. The model is tested on two real-world data from PeMS [199]. We will first present the analysis of the dataset. Meanwhile, the data preprocessing process will also be presented in Section 4.3.1. In Section 4.3.2, there will be a thorough analysis of the experiment results of our work.

4.1 Performance metrics

To compare the performance between different prediction models, we use three performance metrics to evaluate the prediction results: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-Squared (R^2) which can be expressed as follows:

$$\begin{aligned} MAE &= \frac{1}{n}\sum_{i=1}^{n}|\widehat{Y}_i - Y_i|, \\ RMSE &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2}, \\ R^2 &= 1 - \frac{\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2}{\sum_{i=1}^{n}(\overline{Y} - Y_i)^2} = 1 - \frac{MSE}{Var(Y)}, \end{aligned} \tag{4.1}$$

Each of the above metrics has their own features. MAE shows the difference between the real data and the prediction results, while RMSE expands the error when the difference between the real data and prediction results is bigger. As for R-Squared, we can see it as the comparison between the prediction model and the baseline model. When the value of R-Squared is closer to 1, the accuracy of our model is higher. When the value of R-Squared is 0 the accuracy of our model is equal to the baseline model. If R-Squared is less than 0, it means the model does not learn any linear features from the dataset.

Beside the above metrics, we also focus on the time we spend on training and prediction phases, which is important for implementing a prediction model in the real world. To further evaluate the cost-effectiveness of different methods, we provide a new metric

called Gain, which can be calculated as follows:

$$Gain = \frac{R_N^2 - R_B^2}{T_{N_{Train}} - T_{B_{Train}}} * 100\%, \tag{4.2}$$

where R_N^2, R_B^2 represent the R-Squared value for the new model and benchmark respectively, and $T_{N_{Train}}$, $T_{B_{Train}}$ are the training times for the new model and benchmark, respectively. Normally, we choose the model with the shortest training time as the benchmark. The higher value of Gain means the method is more cost-effective.

4.2 Experiments on One-One traffic prediction model and optimization strategy for training process

4.2.1 Datasets

In our experiment, we have used two datasets, i.e., PEMS-bay and NY-urban. These two datasets contain traffic flow data both collected from real-world datasets, PeMS [199], and NYC Open data [243], respectively; and a short comparison of the two datasets in Table 4.1. These two datasets represent different traffic flow prediction tasks. According to the dataset's time interval, PEMS-bay is used to evaluate the model's performance on short-term traffic prediction tasks, while NY-urban is used for the long-term traffic prediction task. Meanwhile, due to the number of records in the dataset, PEMS-bay can help us see the model's efficiency dealing with a large dataset, and NY-urban can help us compare the models' dependence on the size of the dataset.

Last but not least, these two datasets are collected from different road environments. In Fig. 4.1, we compared the average traffic volume of the 30 detectors in each dataset collected in the first week. As we can see, the overall traffic volume in NY-urban is way more than the traffic volume in PEMS-bay, which means that even though the traffic flow trend seems similar, the changes in NY-urban are more significant in PEMS-bay.

Table 4.1: Comparison on the datasets used in the experiment

	PEMS-Bay	**NY-urban**
Road type	Freeway	Urban road
Time Interval	5-min	1-hour
Record Length	4-month	1-week
Number of Records	1e6	6e3
Number of Detectors	30	
Data Type	Traffic flow	

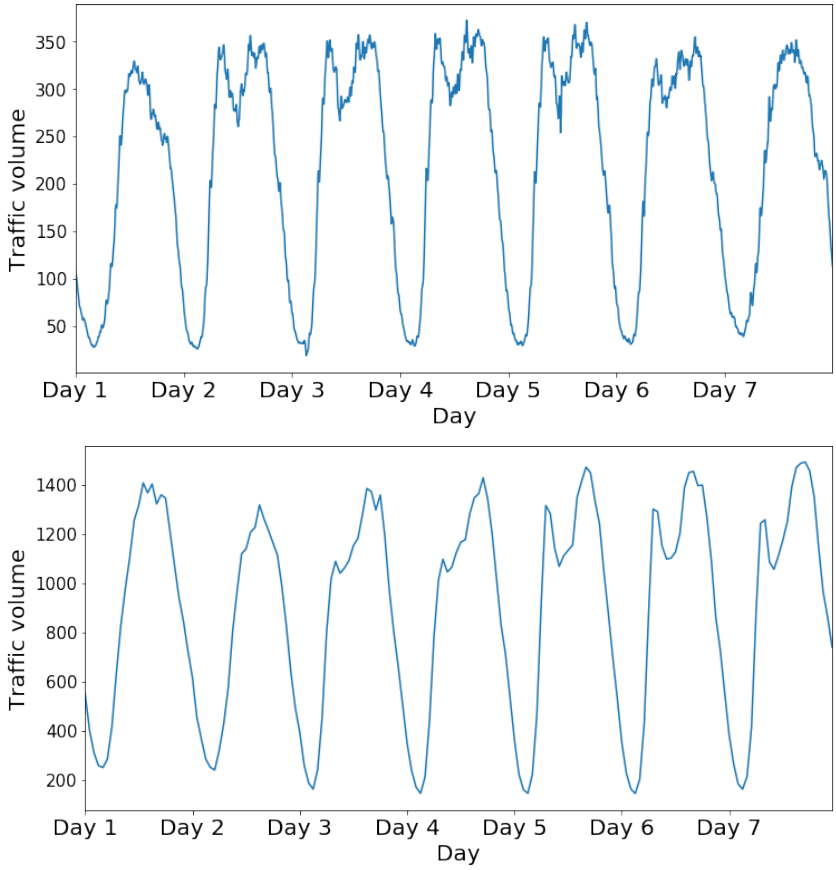


Figure 4.1: Overview on the first week's traffic flow in PEMS-bay (up) and NY-urban (down).

4.2.2 Experiment results

4.2.2.1 Experiment setup

In our experiment, we compared several famous non-parametric models:

- SVM: Linear kernel SVM (L-SVM), RBF kernel SVM (RBF-SVM), and Polynomial kernel SVM (P-SVM)
- FFNN: Single layer FFNN
- RNN: Single layer LSTM and single layer GRU

Furthermore, we analyze the following deep-learning structures for the traffic prediction models:

- FFNN: SAE and DBN
- RNN: Multi-layer RNN, bi-direction RNN, seq2seq RNN, and attention-based seq2seq RNN
- DeepTrend structure

Each model will be trained and tested on both datasets, and all models will use the past five continuous records to predict the traffic flow in the next time interval. There are 70% of the dataset used for training will be randomly shuffled before training the model, and the rest are used for the model test. By random shuffling the dataset, we try to avoid the training inertia caused by the regular vibration in the original dataset. Meanwhile, we use the average of the cross-validation method's results to avoid evaluation deviation.

4.2.2.2 Performances of non-parametric models

First, we will take a look at the performance of each non-parametric models. As we can see in Table 4.2, from a macro perspective, all models perform better on PEMS-bay

Table 4.2: Average experiment results on the two datasets for each non-parametric models.

Models	PEMS-bay					NY-urban				
	RMSE	**MAE**	**R2**	**Training time/s**	**Test time/s**	**RMSE**	**MAE**	**R2**	**Training time/s**	**Test time/s**
L-SVM	16.833387	14.11306	0.982679	2.437065	0.003616	153.756602	119.012254	0.863381	0.043204	0.000115
P-SVM	16.804553	14.071188	0.982709	1.289892	0.004437	153.657461	118.939087	0.863689	0.00742	0.000085
RBF-SVM	21.632685	17.755842	0.970151	0.091694	0.019289	145.859605	116.773003	0.867022	0.000419	0.000113
FFNN	12.129377	8.141788	0.990523	1.830242	0.001698	116.064889	86.491541	0.894535	1.888411	0.000149
LSTM	8.957064	6.123733	0.994689	18.063892	0.003543	117.099682	86.471353	0.899088	3.61464	0.000435
GRU	8.964779	6.150099	0.994679	17.508204	0.003253	113.87133	84.328828	0.902511	3.542995	0.000456

than on NY-urban. All R-square values in PEMS-bay are higher than the same model's R-square value in NY-urban. The main reason is that the size of PEMS-bay is much bigger than NY-urban, which means more data can be used for model training. The models can capture more features in the traffic flow series.

Meanwhile, the RMSE and MAE value in PEMS-bay is much smaller than in NY-urban. It is because that the overall volume in PEMS-bay is smaller than in NY-urban, the same degree of forecast volatility will cause more significant error results in NY-urban. Moreover, the training time is longer when the models are used for PEMS-bay. One reason is that the models need more time to go through all data in the training set. Another reason is that if the training process is too long for a model trained on a small dataset like NY-urban, it is easy to cause an overfitting problem, significantly affecting its accuracy.

Now, let us take one step further, analyzing the performance of each model. As we can see from Table 4.2, the RNN-type models, i.e., LSTM and GRU, have the best accuracy over all of the models. However, LSTM and GRU have slightly different performances on different datasets. LSTM has a better performance than GRU on PEMS-bay, while the models' performance on NY-urban is just the opposite. It means LSTM has a better ability to handle a large dataset than GRU, while the GRU's simpler two-gate structure is beneficial for training on a small dataset. In the meantime, we also notice that GRU's training time is shorter than LSTM's, which is also due to GRU's simpler inner structure.

From Table 4.2, we can see that ML-based methods (FFNN, LSTM, and GRU) have

Table 4.3: Experiment results of different deep-learning structures on the two datasets.

Models		PEMS-bay					NY-urban				
		RMSE	MAE	R2	Training time/s	Test time/s	RMSE	MAE	R2	Training time/s	Test time/s
SAE		10.949906	8.234334	0.991878	11.795200	0.000867	139.889328	105.901834	0.855887	13.721058	0.000686
DBN		10.321363	7.611481	0.993029	24.875117	0.000816	123.099012	90.952072	0.892961	18.422307	0.000639
Dual-layer RNN	LSTM	8.997613	6.129228	0.994709	25.504078	0.004037	120.365883	88.203106	0.885006	4.784408	0.000545
	GRU	8.951501	6.137933	0.994696	24.309060	0.003768	120.634479	88.488250	0.882972	4.723473	0.000512
Triple-layer RNN	LSTM	8.968567	6.164366	0.994671	32.780753	0.004865	122.768399	92.307754	0.876474	5.941472	0.000679
	GRU	8.931924	6.147366	0.994711	30.745161	0.004428	134.661207	101.302997	0.879068	5.709832	0.000627
Bi-drection RNN	LSTM	8.925625	6.113713	0.994709	26.154155	0.003947	128.277454	94.341071	0.876036	4.849733	0.000601
	GRU	8.941581	6.124832	0.994669	24.902018	0.003764	118.797130	86.087539	0.884587	4.682895	0.000528
Seq2seq		10.104538	7.069788	0.993378	20.812815	0.716703	173.877403	135.734316	0.771880	9.549930	0.002336
Attention-based seq2seq		10.520587	7.533946	0.992741	85.711398	1.213986	218.344456	175.590382	0.868432	20.317136	0.003916
DeepTrend		9.730399	6.670463	0.993728	69.398025	0.006623	-	-	-	-	-

better accuracy than the traditional non-parametric model, i.e., SVM. The ML-based models have advantages in extracting complicated pattern relationships from historical data. However, we also notice that the traditional non-parametric model's training time is much shorter than the training time for ML-based methods, a useful feature for real-world implementation. It is because the scale of the inner parameters in an ML-based model is much bigger than a non-parametric model, which will take more time to train the model. Moreover, as we can see that RBF-SVM spend the shortest time to converge in both datasets, the other two types of SVM have higher accuracy on PEMS-bay, and NY-urban just the opposite, which means RBF kernel if not fit for the task on a large dataset. Another thing is that although the training time is shorter for SVM compared to ML-based models, the tensor-based implementation environment for ML-based models reduces the prediction time of the model.

4.2.2.3 Performances of deep-learning structures

In this section, we will discuss the performance of different deep-learning structures. We have to point out that the implementation requirements of the DeepTrend structure, a week's historical data need to be used to calculate the historical average trend. However, due to the NY-urban dataset's size, which only has one-week records, it is not suitable for testing the DeepTrend structure, so there is a missing experiment result for the DeepTrend structure on NY-Urban.

Let us start with the analysis of RNN-related deep-learning structures' performance.

As we can see from Table 4.3, multi-layer RNN structure, including Dual- and Triple-layer RNN structure, and bi-direction RNN structure outperform other deep-learning structures on PEMS-bay. In contrast, the DBN structure outperforms others on NY-urban. It shows us that the deep-learning structure for RNN is more suit for large data traffic prediction tasks, while DBN is relatively more suitable for a small dataset. While we have a close look at the experiment results of the multi-layer RNN structures, we can see that although the number of layers is increasing, the model's accuracy does not have a significant improvement. In the meantime, the model's accuracy is decreasing on NY-urban while the layer is increasing. The reason is that although a more complex model structure will learning more features from historical data, the model is also easier to overfit the training dataset, especially for a small dataset. We also notice that, while the R-square value of dual-layer RNN is very close to bi-direction RNN, the overall error of bi-direction RNN is lower than dual-layer RNN. It means the bi-direction RNN has better performance while the traffic volume is large such as in peak hour. As we can also see from Table 4.3, although there is the same number of layers of RNN in dual-layer RNN, bi-direction RNN, and seq2seq, the performance of the models are very different from each other, especially on NY-urban. The seq2seq structure has a more significant overfitting problem compared to others. Meanwhile, the performance of attention-based seq2seq has a similar performance than the structure without the attention layer on PEMS-bay, while the attention layer can highly improve the model's accuracy on NY-urban.

After discussing the performance on deep-learning structures for RNN, let us take a look at the other deep-learning structures. As we can see in Table 4.3, the DeepTrend structure outperforms SAE and DBN. However, the training time of DeepTrend is much more than SAE and DBN because of the more complicated model structure. Meanwhile, we also notice that, although DBN and SAE use the pre-training strategy, SAE converges faster than DBN, according to the training time of the models. Nevertheless, in the

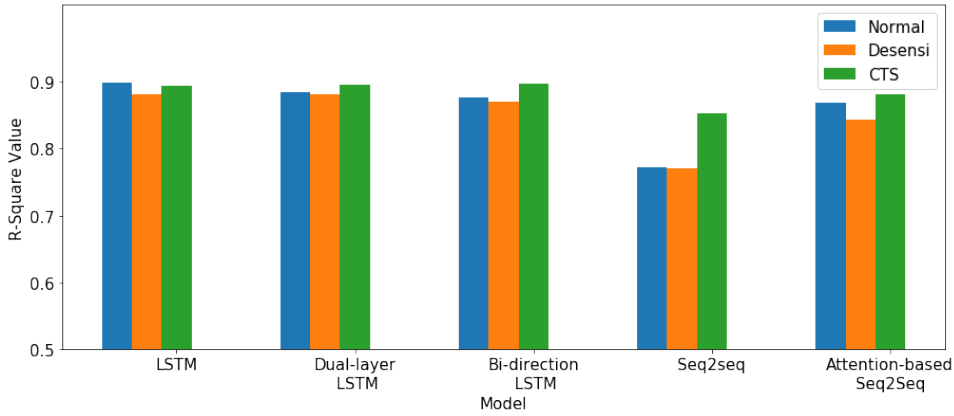


Figure 4.2: Comparison of the R-Square values of models using different training strategies on NY-Urban dataset

meantime, the DBN structure can bring more accuracy to the base model.

By comparing Table 4.2 and Table 4.3, the training time is much longer while using a deep-learning structure than using the base model, especially for FFNN. The reason is that the fully-connected layer in DBN and SAE requires more weights and biases parameters, which will increase the computational burden. Meanwhile, we also conclude that the deep-learning structure can amplify the features of the base model. By taking LSTM and GRU as an example, the differences between LSTM and GRU's training cost are more significant, while the number of deep-learning structure layers is increasing. However, we can see that, although the overall accuracy is higher while using the deep-learning structure on PEMS-bay, the accuracy of the model drop on NY-urban. This result reminds us that a deep-learning structure is not fit for handing a small training dataset. The converge process of the deep-learning structure model is longer than the base model, requiring more data to train the model. A deep-learning structure model trained by a small dataset with insufficient training epochs will not converge. However, if the training data are used in too many epochs, the model will overfit the training dataset, which will also affect the model's accuracy.

Table 4.4: Comparison of the training time of the model by using different training strategies on NY-Urban dataset. (D-LSTM represents dual-layer LSTM, B-LSTM represents bi-direction LSTM, S2S, represents seq2seq structure model, and AS2S means attention-based seq2seq structure)

	Normal/s	**Desensi/s**	**CTS/s**
LSTM	3.614640	1.165748	1.733389
D-LSTM	4.784408	1.536375	1.784391
B-LSTM	4.849733	1.669032	1.809935
S2S	9.549930	1.249903	5.593644
AS2S	20.317136	2.037045	6.203454

4.2.2.4 Performance of optimization algorithms

As we can see from Table 4.2, the RNN model costs at least two times longer than the other base model. It will highly affect the system's overall implementation cost, while the number of detectors is huge. Meanwhile, in Table 4.3, we noticed that the DL structure ML-based models have difficulty improving the base model's performance when there are no adequate training records. It is vital for us to reduce the implementation time for the RNN-based model and its DL structure models.

In this section, we will focus on the two optimization algorithms' performance, i.e., Desensitization algorithm and CTS. We will test the training algorithm using the NY-urban dataset for both the RNN model (LSTM) and its DL structures. In Table 4.4 and Fig. 4.2, "Normal" means the models are trained by the traditional Back-Propagation method, while "Desensi" means the models are trained using the Desensitization strategy, and "CTS" means the models are trained using CTS.

As we can see from Table 4.4, both Desensitization strategy and CTS can highly reduce the model's training time. However, due to the more complex structure, CTS's average training time is higher than the Desensi strategy. Even so, the training time for models training with CTS is at least close to half of the models' training time using regular training strategy.

Meanwhile, according to Fig. 4.2, using either of the two optimization algorithms can

Table 4.5: Comparison of the models' RMSE value under high-volume traffic situations while using different training strategies on NY-Urban dataset.

	Normal	**Desensi**	**CTS**
LSTM	**136.498583**	144.956763	139.182466
D-LSTM	**128.781329**	132.596861	142.277346
B-LSTM	135.541532	**131.664661**	141.005951
S2S	216.750922	226.531404	**172.365872**
AS2S	**175.594376**	184.266120	184.843788

obtain acceptable accuracy. By using Desensitization, the model is trained on the base dataset first, which will help the model learn the overall data features. Meanwhile, on the second broadcast stage, the model is trained on different sub-datasets to learn specific traffic pattern features of a given detector, which will help the model avoid overfitting problem.

Meanwhile, although the CTS takes a longer time to train the model, its overall accuracy is better than Desensi. Except for LSTM, the accuracy drops a little, the other models all have better performance than the situation training in the normal way. CTS process assures the pre-trained model's qualification on basic feature extraction by embedding the Desensitization process. Then, the model refinement process further improves the accuracy of the model. Meanwhile, the three verification stage reduce the model's overfitting risk. CTS improves the model's efficiency without compromising the model's accuracy, which is significantly improved compared to the Desensi strategy.

Still, these two optimization methods has a same disadvantage. As we can see from Table 4.5, the models' performance drops during high-volume situation. The main reason is that the pre-trained model is trained by using historical average data on each detector, which will inevitably lead the model to bias towards conservative predictions on high-volume situation.

4.3 Experiments on proposed hybrid ml-based prediction system for Multi-Multi traffic prediction tasks

4.3.1 Data preparation

The vehicular traffic flow data we use is collected by using loop detector on two large-scale traffic networks from PeMS [199], i.e., METR-LA and PEMS-BAY. The detectors' geographical information is provided in [225]. PeMS collected the real-time from more than 39,000 individual detectors on freeway in California. It provides us the information including flow, occupancy, speed and other useful data of each lane for each detector point. Meanwhile, the raw time interval we collected from PEMS is set to be 5 minutes which contains more nonlinear features, and also gives us more space for subsequent data processing. The data of the above two road networks are both from the highway. There are 200 sensors in METR-LA network, which are located in Los Angeles County. As for PEMS-BAY, there are 300 sensors located in the Bay area of California. Both of these two datasets were collected between April 1, 2018 and July 31, 2018. To have an intuitive impression of these two datasets, we take the data of the first week from all detectors, and calculate the mean value on each timestamp. As we can see from Fig. 4.3, the peak road traffic volume on METR-LA is much higher than on PEMS-BAY, as well as the vehicular traffic volume on each timestamp. Meanwhile, we can see that the two peak-time vibrations on weekdays are more apparent in PEMS-BAY.

The weighted adjacency matrix is built by using threshold Gaussian kernel method according to [244] based on the distance between each other. The raw data is smoothed by using moving average, and the window size is set to be 3. As shown in Figure 4.4, we can see that when the size of smooth window is set to be 3, we can smooth the raw data without losing a lot of useful non-seasonal vibrations. After the data is smoothed, it is normalized by MinMaxScaler before inputted into the prediction models. 7/10 of

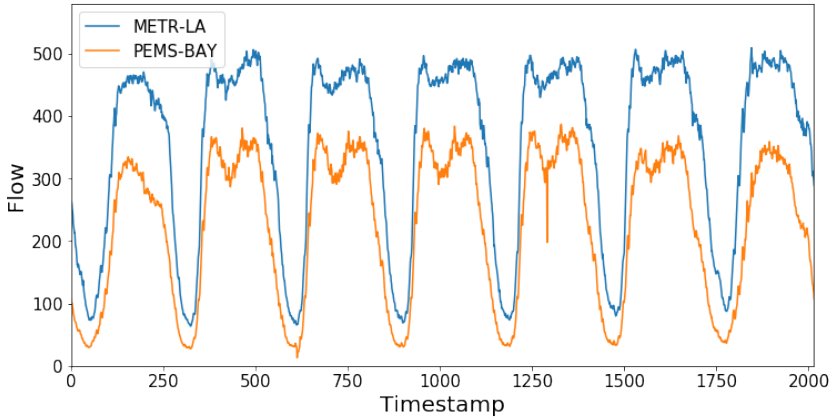


Figure 4.3: Mean value over all detectors in two datasets in the first week of April (Sunday to Saturday).

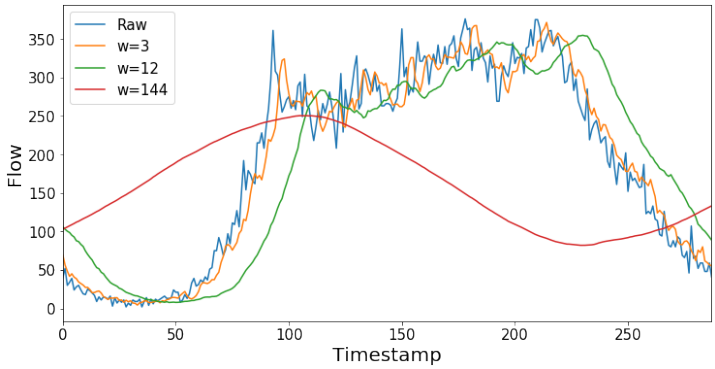


Figure 4.4: Comparison of the data smoothed with different window size from April 2 on detector 400001

the data is used for training the prediction models, which are shuffled randomly to avoid the overfitting situation and to ensure the generalization of the models; 2/10 of the data is used for validation; and the rest of the data is used for model testing. As for the evaluation of the parallel strategy, half of the validation data is used for validation tasks, while the remaining half of the data is used for fine-tuning the model parameters.

In the remainder of the section, we will analyze the performance of the prediction system we proposed by using the real-world data records.

Table 4.6: Accuracy, time consumption and cost-effectiveness for all models on two real-world datasets

Models	PEMS-BAY					METR-LA				
	MAE	**RMSE**	**R2**	T_{train}**(s)**	**Gain**	**MAE**	**RMSE**	**R2**	T_{train}**(s)**	**Gain**
S2S	22.16	35.59	0.9423	93	-	27.41	38.65	0.9567	85	-
SA2S	25.31	41.75	0.9201	127	-6.53e-4	29.70	43.19	0.9456	145	-1.84e-4
DARNN	22.05	36.81	0.9408	369	-5.43e-6	25.99	37.61	0.9592	307	1.13e-5
DCRNN	20.11	31.24	0.9574	3640	4.26e-6	22.97	32.55	0.9695	3449	3.81e-6
M1	21.23	34.88	0.9477	2352	2.39e-6	24.489	34.99	0.9643	2230	3.54e-6
M2	17.72	28.22	0.9657	367	8.54e-5	20.74	29.20	0.9751	350	6.94e-5

4.3.2 Performance of the prediction model

4.3.2.1 Models and experiment environment setting

The models we selected in our work are as follows:

- S2S: Sequence to sequence model using GRU kernel [161].
- SA2S: Sequence to sequence model using GRU kernel with Attention layer in encoder.
- DARNN: Dual-stage Attention-based recurrent neural network provided by Qin et al. [245]
- DCRNN: Diffusion convolutional recurrent neural network provided by Li et al. [225]
- M1: The model we proposed with three-layer GCN in the first part of the model.
- M2: The model we proposed with Diffusion GCN in the first part of the model.

All models are tested on the same hardware environment (System: Ubuntu 16.04 LTS, CPU: I7-8700k, GPU: GeForce GTX 1080 Ti).

4.3.2.2 Accuracy and efficiency of the models

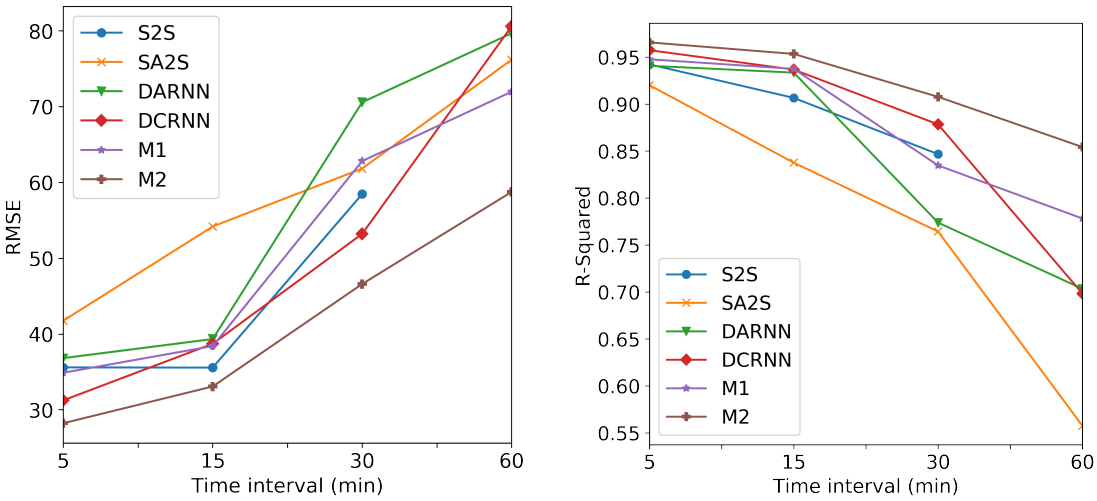
The results of the experiments are shown in Table 4.6. Generally speaking, the training time of the models trained for PEMS-BAY is longer than the models trained for METR-LA. This is because the size of the METR-LA dataset is smaller than that of PEMS-BAY.

Meanwhile, from the value of R2 we can see that the accuracy of the models trained for METR-LA is higher than the models for PEMS-BAY. We mainly believe that is because the vibration during the daytime in PEMS-BAY is more abundant than in METR-LA, which is more difficult to fit.

Now, let us focus on the performance of different models. The first thing is that the model we proposed with diffusion GCN in front of the Attention-based sequence to sequence structure outperforms other models in any dataset in terms of accuracy, time efficiency and cost-effectiveness. By comparing M1 and M2, we can see that diffusion GCN is more effective and efficient than multi-layer GCN. By comparing M1, M2 and SA2S, we can see that the spatial extraction phase is a valid way to improve the accuracy of the prediction model. By comparing models S2S, SA2S, and DARNN, we noticed that the complexity of the model does not always have a positive affect on its accuracy, while the time consumption normally increases in congruence with the complexity of model. Another point is that models' performance varies when applied to different datasets. For example, the Gain value of DARNN is relatively bigger than other models except M2 on METR-LA; however, the Gain value is negative when applied to PEMS-BAY. This is related to the complexity of the dataset, where PEMS-BAY has more complicated non-linear features than METR-LA, as we addressed in the first part of this section.

4.3.2.3 Online prediction performance

In this part, we will focus on the accuracy of the models for online prediction tasks. The prediction interval is set to be 15 min, 30 min, and 1 hour; meanwhile, we choose PEMS-BAY as the test dataset. The performance of online predictions for each model without using a refinement layer is shown in Fig. 4.5. As we can see from the picture, the accuracy of the model deteriorates, when the prediction interval increases. The models we propose, i.e., M1 and M2, show better performance when the prediction interval increases, particularly when the prediction interval is 60 min. Meanwhile, M2 retains the lowest



(a) RMSE (b) R-Squared

Figure 4.5: The online prediction results of the models without using the refinement learning method. Here, R2 value that is less than 0 or RMSE bigger than 150 is omitted.

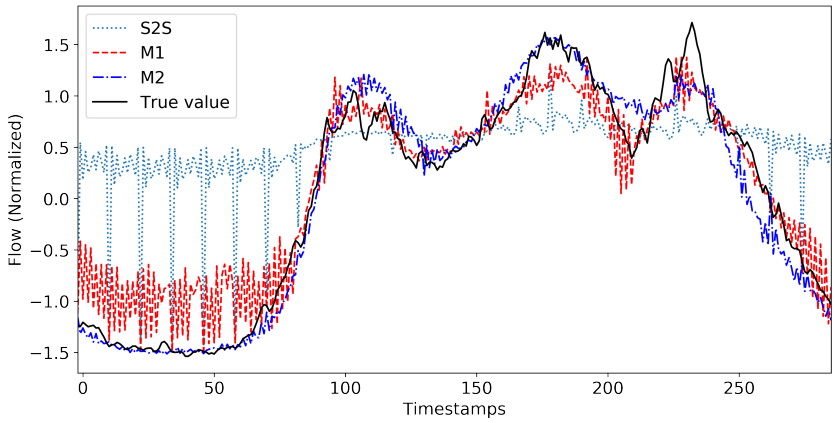


Figure 4.6: Prediction results for July 31 by S2S, M1, and M2 while the prediction interval is set to be 60 min

accuracy deterioration speed, which indicates the model is more stable than the other models; in another words, the model has better robustness. One thing we also notice is that although S2S shows strong prediction results when the prediction interval is small, the model's performance becomes unacceptable when the interval is larger. The Fig. 4.6 gives us an intuitive expression of what has happened to the prediction models when the prediction interval becomes bigger. As we can see, all prediction models have more incorrect vibrations, especially when the vehicular traffic flow is low. The model tends to give a much bigger value when the current vehicular traffic volume is small, partly because the model is too sensitive to the vibration. While the current vehicular traffic volume is small, the vibration caused by the small prediction loss has bigger influence compared to the time when the vehicular traffic volume is bigger. This negative influence will be accumulated until the next data update, which will lead to a large prediction error. As we can see, the S2S almost lost all its seasonal features; meanwhile, M1 and M2 can make a reasonable prediction during the peak hours. Besides, we also notice that M2 can keep a relatively stable prediction performance when the vehicular traffic volume is small.

To further improve the robustness of the model, we apply the refinement learning strategy to the models. Here, we take M2 as an example, and we use a single-layer FFNN as the refinement layer. The comparison of online prediction results between the model using refinement learning and what the model does not use is shown in Table 4.7. From the Table, we can see that the refinement learning strategy can highly improve the robustness of the prediction model, and make the prediction results much more stable than the traditional method. Meanwhile, according to the value of Gain, we can see that the refinement learning is more useful when the prediction interval is bigger.

Table 4.7: The comparison of the model using refinement strategy and the model that does not. Each value of R2 and RMSE represents the value of the model not using refinement learning, and the model that does, respectively.

	15-min	**30-min**	**60-min**
R2	0.9534/0.9607	0.9077/0.9606	0.8543/0.961
RMSE	33.07/30.15	46.59/30.14	58.75/29.98
T	59	114	216
Gain	1.23e-4	4.64e-4	4.93e-4

Table 4.8: Prediction results of M2 using parallel training strategy and M2 using independent training, where $T_{c_{Train}}$ is the time we spend training the central system, and $T_{s_{Train}}$ is the time we spend updating and training the parameters on one sub-system.

Model		**Parallel**	**Separated**
PEMS-BAY	**MAE**	12.16	11.47
	RMSE	18.91	17.98
	R2	0.9867	0.9860
	$T_{c_{Train}}$	353	-
	$T_{s_{Train}}$	50	261
METR-LA	**MAE**	14.38	14.71
	RMSE	20.36	20.97
	R2	0.9881	0.9873
	$T_{c_{Train}}$	367	-
	$T_{s_{Train}}$	35	185

4.3.2.4 Parallel training strategy performance

In this part, we compared the parallel training strategy with independent training strategy. For independent training strategy, i.e., the traditional training method, each subsystem kept their own parameters and does not get pre-trained model parameters from the central system. As can be seen from Table 4.8, the parallel training strategy perform as well as the independent training strategy. In addition, the model using parallel training strategy can update the parameters in a really short time, giving us the opportunity to update the parameters of the prediction system based on the more recent vehicular traffic records.

Chapter 5

Conclusions

In our work, firstly, we reviewed and tested several famous non-parametric traffic prediction models and their deep-learning structures on traffic prediction tasks both on the freeway and urban areas for One-One traffic prediction tasks. Our experiment compared the accuracy and the implementation time cost for the base models and the deep-learning structures. From the experiment results, we can see that ML-based models and their deep-learning structure have an advantage in prediction accuracy, especially while the model has sufficient training data. However, the traditional non-parametric model, such as SVM, has advantages on implementation cost. We need to choose a suitable model

according to the size of the dataset. Moreover, we also introduced a new optimization strategy, i.e., CTS, for the traditional training process. In the experiment, we proved that the CTS has a better overall performance than the Desensitization algorithm. We proved that CTS could reduce the training time for the whole prediction system and improve the model's accuracy, significantly improved compared to the Desensitization algorithm.

Secondly, we provided a new hybrid deep learning model integrating GCN with Attention-based sequence to sequence model with bi-directional GRU kernel. Meanwhile, we proposed a new performance metric called Gain, which can help us evaluate the cost-effectiveness of the new methods. We also introduced a new online training strategy by using refinement learning. Furthermore, we provided a training strategy that could further use the vehicular network structure to improve the efficiency and accuracy of the model. Using the new training strategy, we can easily update the model by using the most recent records without losing the seasonal features. According to our experiments, we have proved that the model has good accuracy and efficiency. In the meantime, we have proved that the model has high robustness when facing online prediction tasks. We also provided the efficiency of the new prediction strategy and the training strategy.

There are many possible research directions in the future.

Start with the Desensitization and CTS training strategy. It is essential to provide a more thorough analysis for Desensitization based on experiments to see whether the method can maintain the high performance for a more complicated model, such as a Deep-Learning model. Furthermore, we want to evaluate how data loss affects the performance of the training strategy.

For large transportation network prediction, it is interesting to seek the combination of our model with our ITS applications, such as VC or congestion control system, to see how our system can improve their performance. Secondly, it is essential to consider the spatial relationship change at different times of the day and find a way to use this

feature. Moreover, we would like to provide models for traffic prediction on the combined transportation network, such as freeway and urban road. Last but not least, we think that it is necessary to reduce the prediction interval further so that we can include the accident situation on the road and provide more useful information for road management.

References

[1] Hannah Ritchie and Max Roser. "Urbanization". In: *Our World in Data* (2020). https://ourworldindata.org/urbanization.

[2] Emmanuel Dechenaux, Shakun D. Mago, and Laura Razzolini. "Traffic congestion: an experimental study of the Downs-Thomson paradox". In: *Experimental Economics* 17.3 (Sept. 2014), pp. 461–487.

[3] Maram Bani Younes and Azzedine Boukerche. "Intelligent traffic light controlling algorithms using vehicular networks". In: *IEEE transactions on vehicular technology* 65.8 (2015), pp. 5887–5899.

[4] Peng Sun, Noura AlJeri, and Azzedine Boukerche. "DACON: A novel traffic prediction and data-highway-assisted content delivery protocol for intelligent vehicular networks". In: *IEEE Transactions on Sustainable Computing* 5.4 (2020), pp. 501–513.

[5] Noura Aljeri and Azzedine Boukerche. "An Adaptive Traffic-Flow based Controller Deployment Scheme for Software-Defined Vehicular Networks". In: *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 2020, pp. 191–198.

[6] Osama Abumansoor and A. Boukerche. "A Secure Cooperative Approach for Nonline-of-Sight Location Verification in VANET". In: *IEEE Transactions on Vehicular Technology* 61 (2012), pp. 275–285.

[7] Fabricio A Silva et al. "Vehicular networks: A new challenge for content-delivery-based applications". In: *ACM Computing Surveys* 49.1 (2016), pp. 1–29.

[8] Clayson Celes et al. "Improving vanet simulation with calibrated vehicular mobility traces". In: *IEEE Transactions on Mobile Computing* 16.12 (2017), pp. 3376–3389.

[9] Noura Aljeri and Azzedine Boukerche. "Mobility and handoff management in connected vehicular networks". In: *Proceedings of the 16th ACM International Symposium on Mobility Management and Wireless Access*. 2018, pp. 82–88.

[10] Peng Sun, Noura AlJeri, and Azzedine Boukerche. "An energy-efficient proactive handover scheme for vehicular networks based on passive rsu detection". In: *IEEE Transactions on Sustainable Computing* 5.1 (2018), pp. 37–47.

[11] Azzedine Boukerche, Alexander Magnano, and Noura Aljeri. "Mobile IP handover for vehicular networks: Methods, models, and classifications". In: *ACM Computing Surveys* 49.4 (2017), pp. 1–34.

[12] Noura Aljeri, Mohammed Almulla, and Azzedine Boukerche. "An efficient fault detection and diagnosis protocolfor vehicular networks". In: *Proceedings of the third ACM international symposium on Design and analysis of intelligent vehicular networks and applications*. 2013, pp. 23–30.

[13] Noura Aljeri and Azzedine Boukerche. "An efficient handover trigger scheme for vehicular networks using recurrent neural networks". In: *Proceedings of the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*. 2019, pp. 85–91.

[14] Noura Aljeri and Azzedine Boukerche. “Fog-enabled vehicular networks: A new challenge for mobility management”. In: *Internet Technology Letters* 3.6 (2020), e141.

[15] Noura Aljeri and Azzedine Boukerche. “A dynamic MAP discovery and selection scheme for predictive hierarchical MIPv6 in vehicular networks”. In: *IEEE Transactions on Vehicular Technology* 69.1 (2019), pp. 793–806.

[16] Noura Aljeri and Azzedine Boukerche. “Mobility Management in 5G-enabled Vehicular Networks: Models, Protocols, and Classification”. In: *ACM Computing Surveys* 53.5 (2020), pp. 1–35.

[17] Noura Aljeri and Azzedine Boukerche. “A Performance Evaluation of Time-Series Mobility Prediction for Connected Vehicular Networks”. In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. 2020, pp. 127–131.

[18] A. Boukerche et al. “Localization systems for wireless sensor networks”. In: *IEEE Wireless Communications* 14.6 (2007), pp. 6–12.

[19] Y. Ren et al. “Monitoring patients via a secure and mobile healthcare system”. In: *IEEE Wireless Communications* 17.1 (2010), pp. 59–65.

[20] Azzedine Boukerche et al. “Secure localization algorithms for wireless sensor networks”. In: *IEEE Communications Magazine* 46.4 (2008), pp. 96–101.

[21] Azzedine Boukerche and Yonglin Ren. “A trust-based security system for ubiquitous and pervasive computing environments”. In: *Computer communications* 31.18 (2008), pp. 4343–4351.

[22] Rodolfo WL Coutinho et al. “GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks”. In: *Proceedings of the IEEE International Conference on communications*. 2014, pp. 251–256.

[23] Azzedine Boukerche and Yonglin Ren. "A secure mobile healthcare system using trust-based multicast scheme". In: *IEEE Journal on Selected Areas in Communications* 27.4 (2009), pp. 387–399.

[24] Azzedine Boukerche et al. "An artificial immune based intrusion detection model for computer and telecommunication systems". In: *Parallel Computing* 30.5-6 (2004), pp. 629–646.

[25] Azzedine Boukerche et al. "DV-Loc: a scalable localization protocol using Voronoi diagrams for wireless sensor networks". In: *IEEE Wireless Communications* 16.2 (2009), pp. 50–55.

[26] Azzedine Boukerche, Xin Fei, and Regina B Araujo. "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange". In: *Computer Communications* 30.14-15 (2007), pp. 2708–2720.

[27] Azzedine Boukerche et al. "An agent based and biological inspired real-time intrusion detection and security model for computer network operations". In: *Computer Communications* 30.13 (2007), pp. 2649–2660.

[28] Azzedine Boukerche and Xu Li. "An agent-based trust and reputation management scheme for wireless sensor networks". In: *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.* Vol. 3. 2005, 5–pp.

[29] Rodolfo WL Coutinho et al. "Design guidelines for opportunistic routing in underwater networks". In: *IEEE Communications Magazine* 54.2 (2016), pp. 40–48.

[30] Richard WN Pazzi and Azzedine Boukerche. "Mobile data collector strategy for delay-sensitive applications over wireless sensor networks". In: *Computer Communications* 31.5 (2008), pp. 1028–1039.

[31] Azzedine Boukerche and Xin Fei. "A voronoi approach for coverage protocols in wireless sensor networks". In: *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*. 2007, pp. 5190–5194.

[32] Sergio Correia, Azzedine Boukerche, and Rodolfo I Meneguette. "An architecture for hierarchical software-defined vehicular networks". In: *IEEE Communications Magazine* 55.7 (2017), pp. 80–86.

[33] Amir Darehshoorzadeh and Azzedine Boukerche. "Underwater sensor networks: A new challenge for opportunistic routing protocols". In: *IEEE Communications Magazine* 53.11 (2015), pp. 98–107.

[34] Richard W Pazzi and Azzedine Boukerche. "Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 11.1 (2014), pp. 1–22.

[35] Robson E De Grande and Azzedine Boukerche. "Dynamic balancing of communication and computation load for HLA-based simulations on large-scale distributed systems". In: *Journal of Parallel and Distributed Computing* 71.1 (2011), pp. 40–52.

[36] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. "An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks". In: *Mobile Networks and Applications* 13.6 (2008), pp. 614–626.

[37] Azzedine Boukerche and Sotiris Nikoletseas. "Protocols for data propagation in wireless sensor networks". In: *Wireless communications systems and networks*. 2004, pp. 23–51.

[38] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. "A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range". In: *Computer communications* 29.4 (2006), pp. 477–489.

[39] Azzedine Boukerche and Damla Turgut. "Secure time synchronization protocols for wireless sensor networks". In: *IEEE Wireless Communications* 14.5 (2007), pp. 64–69.

[40] Azzedine Boukerche, Richard WN Pazzi, and Jing Feng. "An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks". In: *Computer Communications* 31.11 (2008), pp. 2716–2725.

[41] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. "A real-time lane marking localization, tracking and communication system". In: *Computer Communications* 73 (2016), pp. 132–143.

[42] Azzedine Boukerche et al. "A reliable synchronous transport protocol for wireless image sensor networks". In: *2008 IEEE Symposium on Computers and Communications*. 2008, pp. 1083–1089.

[43] Renfei Wang et al. "LIAITHON: A location-aware multipath video streaming scheme for urban vehicular networks". In: *2012 IEEE Symposium on Computers and Communications (ISCC)*. 2012, pp. 436–441.

[44] Baraq Ghaleb et al. "A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations". In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1607–1635.

[45] Hadi Habibzadeh et al. "Sensing, communication and security planes: A new challenge for a smart city system design". In: *Computer Networks* 144 (2018), pp. 163–200.

[46] Azzedine Boukerche and Samer Samarah. "An efficient data extraction mechanism for mining association rules from wireless sensor networks". In: *2007 IEEE International Conference on Communications*. 2007, pp. 3936–3941.

[47] Noura Aljeri and Azzedine Boukerche. “Performance evaluation of movement prediction techniques for vehicular networks”. In: *Proceesdings of the IEEE International Conference on Communications*. 2017, pp. 1–6.

[48] Peng Sun, Noura AlJeri, and Azzedine Boukerche. “A novel passive road side unit detection scheme in vehicular networks”. In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*. 2017, pp. 1–5.

[49] Noura Aljeri et al. “A reliable quality of service aware fault tolerant gateway discovery protocol for vehicular networks”. In: *Wireless Communications and Mobile Computing* 15.10 (2015), pp. 1485–1495.

[50] Noura Aljeri and Azzedine Boukerche. “A predictive collision detection protocol using vehicular networks”. In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE. 2017, pp. 1–5.

[51] Azzedine Boukerche and Alessandro Fabbri. “Partitioning Parallel Simulation of Wireless Networks”. In: *Proceedings of the 32nd Conference on Winter Simulation*. WSC ’00. 2000, pp. 1449–1457.

[52] Azzedine Boukerche et al. “Towards a secure hybrid adaptive gateway discovery mechanism for intelligent transportation systems”. In: *Security and Communication Networks* 9.17 (2016), pp. 4027–4047.

[53] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. “Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.3 (2011), pp. 717–735.

[54] Maram Bani Younes and Azzedine Boukerche. “A performance evaluation of an efficient traffic congestion detection protocol (ECODE) for intelligent transportation systems”. In: *Ad Hoc Networks* 24 (2015), pp. 317–336.

[55] Abdul Jabbar Siddiqui, Abdelhamid Mammeri, and Azzedine Boukerche. "Real-time vehicle make and model recognition based on a bag of SURF features". In: *IEEE Transactions on Intelligent Transportation Systems* 17.11 (2016), pp. 3205–3219.

[56] Maram Bani Younes and Azzedine Boukerche. "An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems". In: *Wireless Networks* 24.7 (2018), pp. 2451–2463.

[57] Narcisse Nya and Bruno Baynat. "Performance Model for 4G/5G Heterogeneous Networks with Different Classes of Users". In: *ACM MSWiM*. 2017, pp. 171–178.

[58] Yunfeng Gu and Azzedine Boukerche. "HD Tree: A Novel Data Structure to Support Multi-Dimensional Range Query for P2P Networks". In: *J. Parallel Distrib. Comput.* 71.8 (Aug. 2011), pp. 1111–1124.

[59] Bruno Olivieri and Markus Endler. "DADCA: An Efficient Distributed Algorithm for Aerial DataCollection from Wireless Sensors Networks by UAVs". In: *ACM MSWiM*. 2017, pp. 129–136.

[60] H.A.B.F. de Oliveira et al. "Directed position estimation: a recursive localization approach for wireless sensor networks". In: *Proceedings. 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.* 2005, pp. 557–562.

[61] Azzedine Boukerche et al. "A Voronoi Approach for Scalable and Robust DV-Hop Localization System for Sensor Networks". In: *2007 16th International Conference on Computer Communications and Networks*. 2007, pp. 497–502.

[62] Azzedine Boukerche, Xin Fei, and Regina B. Araujo. "An Energy Aware Coverage-Preserving Scheme for Wireless Sensor Networks". In: *Proceedings of the 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sen-*

sor, and Ubiquitous Networks. PE-WASUN '05. Montreal, Quebec, Canada, 2005, pp. 205–213.

[63] Maram Bani Younes and Azzedine Boukerche. "Efficient traffic congestion detection protocol for next generation VANETs". In: *2013 IEEE International Conference on Communications (ICC)*. 2013, pp. 3764–3768. DOI: 10.1109/ICC.2013.6655141.

[64] Cristiano Rezende et al. "VIRTUS: A resilient location-aware video unicast scheme for vehicular networks". In: *2012 IEEE International Conference on Communications (ICC)*. 2012, pp. 698–702.

[65] Noura Aljeri et al. "A Performance evaluation of load balancing and QoS-aware gateway discovery protocol for VANETs". In: *2013 27th International Conference on Advanced Information Networking and Applications Workshops*. 2013, pp. 90–94.

[66] Cristiano Rezende et al. "A reactive and scalable unicast solution for video streaming over VANETs". In: *IEEE Transactions on Computers* 64.3 (2014), pp. 614–626.

[67] Azzedine Boukerche and E Robson. "Vehicular cloud computing: Architectures, applications, and mobility". In: *Computer networks* 135 (2018), pp. 171–189.

[68] T. Zhang, R. E. De Grande, and A. Boukerche. "Vehicular Cloud: Stochastic Analysis of Computing Resources in a Road Segment". In: *Proc. ACM PE-WASUM*. 2015, pp. 9–16.

[69] Azzedine Boukerche. *Handbook of algorithms for wireless networking and mobile computing*. CRC Press, 2005.

[70] Azzedine Boukerche and Amir Darehshoorzadeh. "Opportunistic routing in wireless networks: Models, algorithms, and classifications". In: *ACM Computing Surveys* 47.2 (2014), pp. 1–36.

[71] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. "A mobility management scheme for wireless mesh networks based on a hybrid routing protocol". In: *Computer Networks* 54.4 (2010), pp. 558–572.

[72] Thanasis Antoniou et al. "A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range". In: *Proceedings of the 37th Annual Simulation Symposium.* 2004, pp. 43–52.

[73] Azzedine Boukerche et al. "A hardware accelerator for the fast retrieval of DIALIGN biological sequence alignments in linear space". In: *IEEE Transactions on Computers* 59.6 (2010), pp. 808–821.

[74] Vic Barnett. *Environmental statistics: methods and applications.* John Wiley & Sons, 2005.

[75] Noura Aljeri and Azzedine Boukerche. "A Novel Online Machine Learning Based RSU Prediction Scheme for Intelligent Vehicular Networks". In: *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA).* 2019, pp. 1–8.

[76] Noura Aljeri and Azzedine Boukerche. "A two-tier machine learning-based handover management scheme for intelligent vehicular networks". In: *Ad Hoc Networks* 94 (2019), p. 101930.

[77] A. Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks.* John Wiley Sons, 2008.

[78] Azzedine Boukerche et al. "Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems". In: *Computer Communications* 31.12 (2008), pp. 2838–2849.

[79] A. Boukerch, L. Xu, and K. EL-Khatib. "Trust-based security for wireless ad hoc and sensor networks". In: *Computer Communications* 30.11 (2007), pp. 2413–2427.

[80] A. Boukerche et al. “SDAR: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks”. In: *29th Annual IEEE International Conference on Local Computer Networks*. 2004, pp. 618–624.

[81] Azzedine Boukerche and Samer Samarah. “A novel algorithm for mining association rules in wireless ad hoc sensor networks”. In: *IEEE Transactions on Parallel and Distributed Systems* 19.7 (2008), pp. 865–877.

[82] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. “Analysis of a randomized congestion control scheme with DSDV routing in ad hoc wireless networks”. In: *Journal of Parallel and Distributed Computing* 61.7 (2001), pp. 967–995.

[83] Azzedine Boukerche et al. “An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks”. In: *computer communications* 28.10 (2005), pp. 1193–1203.

[84] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. “A distributed fault identification protocol for wireless and mobile ad hoc networks”. In: *Journal of parallel and distributed computing* 68.3 (2008), pp. 321–335.

[85] Azzedine Boukerche and Xin Fei. “A coverage-preserving scheme for wireless sensor network with irregular sensing range”. In: *Ad hoc networks* 5.8 (2007), pp. 1303–1316.

[86] Yonglin Ren and Azzedine Boukerche. “Modeling and managing the trust for wireless and mobile ad hoc networks”. In: *2008 IEEE International Conference on Communications*. 2008, pp. 2129–2133.

[87] Azzedine Boukerche. “A simulation based study of on-demand routing protocols for ad hoc wireless networks”. In: *Proceedings. 34th Annual Simulation Symposium*. 2001, pp. 85–92.

[88] Athanasios Bamis et al. “A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks”. In: *Computer Networks* 52.1 (2008), pp. 130–154.

[89] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. “Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols”. In: *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*. 2006, pp. 18–27.

[90] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. “Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages”. In: *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. 2009, pp. 1–6.

[91] Renê Oliveira et al. “Reliable data dissemination protocol for VANET traffic safety applications”. In: *Ad Hoc Networks* 63 (2017), pp. 30–44.

[92] Azzedine Boukerche and Steve Rogers. “Performance of GZRP ad hoc routing protocol”. In: *Journal of Interconnection Networks* 2.01 (2001), pp. 31–48.

[93] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. “Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks”. In: *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*. 2006, pp. 165–172.

[94] Azzedine Boukerche and Yonglin Ren. “A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks”. In: *Proceedings of the 5th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. 2008, pp. 88–95.

[95] Rodolfo WL Coutinho et al. “A novel void node recovery paradigm for long-term underwater sensor networks”. In: *Ad Hoc Networks* 34 (2015), pp. 144–156.

[96] Cristiano Rezende et al. “A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection”. In: *Ad Hoc Networks* 17 (2014), pp. 1–17.

[97] Azzedine Boukerche, Anis Zarrad, and Regina Araujo. "A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks". In: *IEEE Transactions on Parallel and Distributed Systems* 21.7 (2009), pp. 911–924.

[98] Noura AlJeri and Azzedine Boukerche. "An efficient movement-based handover prediction scheme for hierarchical mobile IPv6 in VANETs". In: *Proceedings of the 15th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2018, pp. 47–54.

[99] Noura Aljeri and Azzedine Boukerche. "An optimized link duration-based mobility management scheme for connected vehicular networks". In: *Proceedings of the 16th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2019, pp. 7–14.

[100] Noura Aljeri and Azzedine Boukerche. "ADVICE-LOC: An adaptive vehicle-centric location management scheme for intelligent connected cars". In: *Ad Hoc Networks* 107 (2020), p. 102223.

[101] Maya Okawa, Hideaki Kim, and Hiroyuki Toda. "Online Traffic Flow Prediction Using Convolved Bilinear Poisson Regression". In: *Mobile Data Management (MDM), 2017 18th IEEE International Conference on*. IEEE. 2017, pp. 134–143.

[102] Pinlong Cai et al. "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting". In: *Transportation Research Part C: Emerging Technologies* 62 (2016), pp. 21–34.

[103] W. Huang et al. "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning". In: *IET Intel. Trans. Syst.* 15.5 (2014), pp. 2191–2201.

[104] Jingyan Guo, Zijun Wang, and Huawei 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. IIP'17. Bangkok, Thailand, 2017, 11:1–11:6.

[105] Zhixiang He, Chi-Yin Chow, and Jia-Dong Zhang. "STANN: A Spatio–Temporal Attentive Neural Network for Traffic Prediction". In: *IEEE Access* 7 (2018), pp. 4795–4806.

[106] Xianglong Luo et al. "Spatiotemporal traffic flow prediction with KNN and LSTM". In: *Journal of Advanced Transportation* 2019 (2019).

[107] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[108] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning*. Cambridge University Press, 2014.

[109] Jiahao Wang and Azzedine 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.

[110] Peng Sun, Noura Aljeri, and Azzedine Boukerche. "Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks". In: *IEEE Network* 34.3 (2020), pp. 178–185.

[111] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[112] John Rice and Erik Van Zwet. "A simple and effective method for predicting travel times on freeways". In: *IEEE Transactions on Intelligent Transportation Systems* 5.3 (2004), pp. 200–207.

[113] Xiaoyan Zhang and John A Rice. "Short-term travel time prediction". In: *Transportation Research Part C: Emerging Technologies* 11.3-4 (2003), pp. 187–210.

[114] Antoine G Hobeika and Chang Kyun Kim. "Traffic-flow-prediction systems based on upstream traffic". In: *Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994*. IEEE. 1994, pp. 345–350.

[115] Jaimyoung Kwon, Benjamin Coifman, and Peter Bickel. "Day-to-day travel-time trends and travel-time prediction from loop-detector data". In: *Transportation Research Record: Journal of the Transportation Research Board* 1717 (2000), pp. 120–129.

[116] Xiang Fei, Chung-Cheng Lu, and Ke Liu. "A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction". In: *Transportation Research Part C: Emerging Technologies* 19.6 (2011), pp. 1306–1318.

[117] Li Li et al. "Robust causal dependence mining in big data network and its application to traffic flow predictions". In: *Transportation Research Part C: Emerging Technologies* 58 (2015), pp. 292–307.

[118] Leif E Peterson. "K-nearest neighbor". In: *Scholarpedia* 4.2 (2009), p. 1883.

[119] Brian L Smith, Billy M Williams, and R Keith Oswald. "Comparison of parametric and nonparametric models for traffic flow forecasting". In: *Transportation Research Part C: Emerging Technologies* 10.4 (2002), pp. 303–321.

[120] H Chang et al. "Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences". In: *IET intelligent transport systems* 6.3 (2012), pp. 292–305.

[121] Zuduo Zheng and Dongcai Su. "Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm". In: *Transportation Research Part C: Emerging Technologies* 43 (2014), pp. 143–157.

[122] Dawen Xia et al. "A map reduce-based nearest neighbor approach for big-data-driven traffic flow prediction". In: *IEEE access* 4 (2016), pp. 2920–2934.

[123] Lun Zhang et al. “An improved k-nearest neighbor model for short-term traffic flow prediction”. In: *Procedia-Social and Behavioral Sciences* 96 (2013), pp. 653–662.

[124] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[125] A. Ding, X. Zhao, and L. Jiao. “Traffic flow time series prediction based on statistics learning theory”. In: *Proc. IEEE ITSC*. 2002, pp. 727–730.

[126] L. Vanajakshi and L. R. Rilett. “A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed”. In: *Proc. IEEE IVS*. 2004, pp. 194–199.

[127] Y. Zhang and Y. Xie. “Forecasting of short-term freeway volume with v-support vector machines”. In: *Transp. Res. Rec.* 2024.1 (2007), pp. 92–99.

[128] Manoel Castro-Neto et al. “Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions”. In: *Expert systems with applications* 36.3 (2009), pp. 6164–6173.

[129] Wei-Chiang Hong et al. “Forecasting urban traffic flow by SVR with continuous ACO”. In: *Applied Mathematical Modelling* 35.3 (2011), pp. 1282–1291.

[130] Marco Lippi, Matteo Bertini, and Paolo Frasconi. “Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.2 (2013), pp. 871–882.

[131] M. Dorigo, V. Maniezzo, and A. Colorni. “Ant System: Optimization by a Colony of Cooperating Agents”. In: *Trans. Sys. Man Cyber. Part B* 26.1 (Feb. 1996), pp. 29–41.

[132] Mei Duo et al. "A short-term traffic flow prediction model based on EMD and GPSO-SVM". In: *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE. 2017, pp. 2554–2558.

[133] Jin Wang and Qixin Shi. "Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory". In: *Transportation Research Part C: Emerging Technologies* 27 (2013), pp. 219–232.

[134] Xianglong Luo, Danyang Li, and Shengrui Zhang. "Traffic flow prediction during the holidays based on DFT and SVR". In: *Journal of Sensors* 2019 (2019).

[135] Zhang Mingheng et al. "Accurate multisteps traffic flow prediction based on SVM". In: *Mathematical Problems in Engineering* 2013 (2013).

[136] Simon Oh et al. "Short-term travel-time prediction on highway: a review of the data-driven approach". In: *Transport Reviews* 35.1 (2015), pp. 4–32.

[137] Franck Jabot. "Why preferring parametric forecasting to nonparametric methods?" In: *Journal of theoretical biology* 372 (2015), pp. 205–210.

[138] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[139] Feng Jin and Shiliang Sun. "Neural network multitask learning for traffic flow forecasting". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 1897–1901.

[140] B Gültekin Çetiner, Murat Sari, and Oğuz Borat. "A neural network based traffic-flow prediction model". In: *Mathematical and Computational Applications* 15.2 (2010), pp. 269–278.

[141] Kranti Kumar, M Parida, and VK Katiyar. "Short term traffic flow prediction for a non urban highway using artificial neural network". In: *Procedia-Social and Behavioral Sciences* 104 (2013), pp. 755–764.

[142] Nicholas G Polson and Vadim O Sokolov. "Deep learning for short-term traffic flow prediction". In: *Transportation Research Part C: Emerging Technologies* 79 (2017), pp. 1–17.

[143] Yilun Lin et al. "Pattern sensitive prediction of traffic flow based on generative adversarial framework". In: *IEEE Transactions on Intelligent Transportation Systems* 20.6 (2018), pp. 2395–2400.

[144] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[145] Yisheng Lv et al. "Traffic flow prediction with big data: A deep learning approach". In: *IEEE Trans. Intelligent Transportation Systems* 16.2 (2015), pp. 865–873.

[146] Li Deng, Dong Yu, et al. "Deep learning: methods and applications". In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387.

[147] Jiquan Ngiam et al. "Multimodal Deep Learning". In: *ICML*. 2011, pp. 689–696. URL: `https://icml.cc/2011/papers/399_icmlpaper.pdf`.

[148] Shidrokh Goudarzi et al. "Self-organizing traffic flow prediction with an optimized deep belief network for internet of vehicles". In: *Sensors* 18.10 (2018), p. 3459.

[149] Xin-She Yang. "Firefly algorithms for multimodal optimization". In: *International symposium on stochastic algorithms*. Springer. 2009, pp. 169–178.

[150] Lu Zhao et al. "Parallel computing method of deep belief networks and its application to traffic flow prediction". In: *Knowledge-Based Systems* 163 (2019), pp. 972–987.

[151] Yanjie Tao, Peng Sun, and Azzedine 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.

[152] Michael I. Jordan. “Artificial Neural Networks”. In: ed. by Joachim Diederich. 1990. Chap. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, pp. 112–127. ISBN: 0-8186-2015-3.

[153] Barak A Pearlmutter. “Learning state space trajectories in recurrent neural networks”. In: *Neural Computation* 1.2 (1989), pp. 263–269.

[154] Axel Cleeremans, David Servan-Schreiber, and James L McClelland. “Finite state automata and simple recurrent networks”. In: *Neural computation* 1.3 (1989), pp. 372–381.

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

[156] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[157] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: (1999).

[158] Felix A Gers and E Schmidhuber. “LSTM recurrent networks learn simple context-free and context-sensitive languages”. In: *IEEE Transactions on Neural Networks* 12.6 (2001), pp. 1333–1340.

[159] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. “Learning precise timing with LSTM recurrent networks”. In: *Journal of machine learning research* 3.Aug (2002), pp. 115–143.

[160] Y. Tian and L. Pan. “Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network”. In: *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. Dec. 2015, pp. 153–158.

[161] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[162] R. Fu, Z. Zhang, and L. Li. “Using LSTM and GRU neural network methods for traffic flow prediction”. In: *Proc. YAC*. 2016, pp. 324–328.

[163] SHI Xingjian et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[164] K. Cho, P. Razvan, and C. Gulcehre. *How to construct deep recurrent neural networks*. [Online] `https://arxiv.org/abs/1312.6026`. Accessed on 2019-10-01.

[165] S. El Hihi and Y. Bengio. “Hierarchical recurrent neural networks for long-term dependencies”. In: *Proc. NIPS*. 1995, pp. 493–499.

[166] M. Hermans and B. Schrauwen. “Training and Analysing Deep Recurrent Neural Networks”. In: *Proc. NIPS*. 2013, pp. 190–198.

[167] Zikai Zou, Peter Gao, and Chang 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.

[168] Hongsuk Yi, Khac-Hoai Nam Bui, and Heejin 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*. WIMS2019. Seoul, Republic of Korea, 2019, 7:1–7:8.

[169] M. Schuster and K. K. Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE Trans. Signal Process.* 45.11 (1997), pp. 2673–2681.

[170] Z. Cui, K. Ruimin, and Y. Wang. “Deep stacked bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction”. In: *Proc. Workshop UrbComp*. 2016.

[171] J. Wang, F. Hu, and L. Li. “Deep Bi-directional Long Short-Term Memory Model for Short-Term Traffic Flow Prediction”. In: *Proc. ICONIP*. 2017, pp. 306–316.

[172] Y. Liu et al. “Short-term traffic flow prediction with Conv-LSTM”. In: *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. Oct. 2017, pp. 1–6.

[173] I Sutskever, O Vinyals, and QV Le. “Sequence to sequence learning with neural networks”. In: *Advances in NIPS* (2014).

[174] Loan NN Do et al. “An effective spatial-temporal attention based neural network for traffic flow prediction”. In: *Transportation Research Part C: Emerging Technologies* 108 (2019), pp. 12–28.

[175] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[176] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*. 2016, pp. 3844–3852.

[177] Bing Yu, Haoteng Yin, and Zhanxing Zhu. “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting”. In: *arXiv preprint arXiv:1709.04875* (2017).

[178] Shengnan Guo et al. “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.

[179] Bing Yu et al. “3D Graph Convolutional Networks with Temporal Graphs: A Spatial Information Free Framework For Traffic Forecasting”. In: *arXiv preprint arXiv:1903.00919* (2019).

[180] Azzedine Boukerche and Jiahao Wang. “A performance modeling and analysis of a novel vehicular traffic flow prediction system using a hybrid machine learning-based model”. In: *Ad Hoc Networks* 106 (2020), p. 102224.

[181] Man-Chun Tan et al. “An aggregation approach to short-term traffic flow prediction”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.1 (2009), pp. 60–69.

[182] Fabio Moretti et al. “Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling”. In: *Neurocomputing* 167 (2015), pp. 3–7.

[183] Yuankai Wu et al. “A hybrid deep learning based traffic flow prediction method and its understanding”. In: *Transportation Research Part C: Emerging Technologies* 90 (2018), pp. 166–180.

[184] Yuanli Gu et al. “An Improved Bayesian Combination Model for Short-Term Traffic Prediction With Deep Learning”. In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[185] Lars Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project”. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[186] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus. “Energy-Aware Data-Centric Routing in Microsensor Networks”. In: *Proceedings of the ACM international workshop on Modeling analysis and Simulation of Wireless Systems*. Association for Computing Machinery, 2003, pp. 42–49.

[187] R. W. L. Coutinho et al. “Geographic and Opportunistic Routing for Underwater Sensor Networks”. In: *IEEE Transactions on Computers* 65.2 (2016), pp. 548–561.

[188] Horacio Antonio Braga Fernandes De Oliveira et al. “An efficient directed localization recursion protocol for wireless sensor networks”. In: *IEEE Transactions on Computers* 58.5 (2008), pp. 677–691.

[189] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. “A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications”. In: *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. 2004, pp. 157–164.

[190] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. “Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments”. In: *Journal of Parallel and Distributed Computing* 66.4 (2006), pp. 586–599.

[191] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. “A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks”. In: *Wireless Networks* 11.5 (2005), pp. 619–635.

[192] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. “A predictive energy-efficient technique to support object-tracking sensor networks”. In: *IEEE Transactions on Vehicular Technology* 60.2 (2010), pp. 656–663.

[193] Horacio ABF Oliveira et al. “Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm”. In: *Performance Evaluation* 66.3-5 (2009), pp. 209–222.

[194] Anahit Martirosyan, Azzedine Boukerche, and Richard Pazzi. “A taxonomy of cluster-based routing protocols for wireless sensor networks”. In: *International*

Symposium on Parallel Architectures, Algorithms, and Networks. 2008, pp. 247–253.

[195] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina B Araujo. "Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol". In: *The IEEE Conference on Local Computer Networks*. 2005, pp. 560–567.

[196] Rodolfo WL Coutinho et al. "Underwater wireless sensor networks: A new challenge for topology control–based systems". In: *ACM Computing Surveys* 51.1 (2018), pp. 1–36.

[197] Horacio ABF Oliveira et al. "Error analysis of localization systems for sensor networks". In: *Proceedings of the 13th annual ACM international workshop on Geographic information systems*. 2005, pp. 71–78.

[198] Ahmed Mostefaoui, Mahmoud Melkemi, and Azzedine Boukerche. "Localized routing approach to bypass holes in wireless sensor networks". In: *IEEE transactions on computers* 63.12 (2013), pp. 3053–3065.

[199] Caltrans. *Caltrans PeMS*. `http://pems.dot.ca.gov/`. Accessed on 2019-10-01.

[200] Muhammad Intizar Ali, Feng Gao, and Alessandra 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.

[201] TDRL and Taek M Kwon. *RTMC traffic dataset*. `http://www.d.umn.edu/~tkwon/TMCdata/TMCarchive.html/`. Accessed on 2019-10-01.

[202] graohhopper. *open-traffic-collection*. `https://github.com/graphhopper/open-traffic-collection/`. Accessed on 2019-10-01.

[203] K Abdelgawad et al. "Advanced traffic simulation framework for networked driving simulators". In: *IFAC PapersOnLine* 49 (2016), pp. 101–108.

[204] Harold J Payne. "FREFLO: A macroscopic simulation model of freeway traffic". In: *Transportation Research Record* 722 (1979).

[205] TransCAD. *TransCAD*. https://www.caliper.com/tcovu.htm. Accessed on 2019-10-01.

[206] inrosoftware. *EMME*. https://www.inrosoftware.com/en/products/emme/. Accessed on 2019-10-01.

[207] Michal Maciejewski. "A comparison of microscopic traffic flow simulation systems for an urban area". In: *Transport Problems* 5 (2010), pp. 27–38.

[208] SUMO. *SUMO*. https://sumo.dlr.de/index.html. Accessed on 2019-10-01.

[209] PTV Vissim. *VISSIM*. https://www.ptvgroup.com/en/solutions/products/ptv-vissim/. Accessed on 2019-10-01.

[210] Trafficware. *SimTraffc*. https://www.trafficware.com/blog/category/simtraffic. Accessed on 2019-10-01.

[211] University of Florida McTrans Center. *TSIS-CORSIM*. http://mctrans.ce.ufl.edu/featured/tsis/. Accessed on 2019-10-01.

[212] Paramics Microsimulation SYSTRA Ltd. *Paramics*. https://www.paramics.co.uk/en/. Accessed on 2019-10-01.

[213] MIT ITSLAB. *MITSIMLAB*. https://its.mit.edu/software/mitsimlab. Accessed on 2019-10-01.

[214] Caliper. *TransModeler*. https://www.caliper.com/transmodeler/default.htm. Accessed on 2019-10-01.

[215] Bailin Yang et al. "Traffic flow prediction using LSTM with feature enhancement". In: *Neurocomputing* 332 (2019), pp. 320–327.

[216] Jintao Ke et al. “Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach”. In: *Transportation Research Part C: Emerging Technologies* 85 (2017), pp. 591–608.

[217] Chenyi Chen et al. “The retrieval of intra-day trend and its influence on traffic prediction”. In: *Transportation research part C: emerging technologies* 22 (2012), pp. 103–118.

[218] Huaxiu Yao et al. “Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction”. In: *arXiv preprint arXiv:1803.01254* (2018).

[219] Junbo Zhang, Yu Zheng, and Dekang Qi. “Deep spatio-temporal residual networks for citywide crowd flows prediction”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[220] Yanjie Tao, Peng Sun, and Azzedine 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.

[221] Yanjie Tao, Peng Sun, and Azzedine 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.

[222] Dawei Chen. “Research on traffic flow prediction in the big data environment based on the improved RBF neural network”. In: *IEEE Transactions on Industrial Informatics* 13.4 (2017), pp. 2000–2008.

[223] Xiaolei Ma et al. “Large-scale transportation network congestion evolution prediction using deep learning theory”. In: *PloS one* 10.3 (2015), e0119044.

[224] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2016. arXiv: `1609.02907 [cs.LG]`.

[225] Yaguang Li et al. “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting”. In: *arXiv preprint arXiv:1707.01926* (2017).

[226] James Atwood and Don Towsley. “Diffusion-convolutional neural networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1993–2001.

[227] Azzedine Boukerche and Sajal K Das. “Dynamic load balancing strategies for conservative parallel simulations”. In: *Proceedings 11th Workshop on Parallel and Distributed Simulation*. 1997, pp. 20–28.

[228] Azzedine Boukerche and Mirela Sechi M Annoni Notare. “Behavior-based intrusion detection in mobile phone systems”. In: *Journal of Parallel and Distributed Computing* 62.9 (2002), pp. 1476–1490.

[229] Azzedine Boukerche and Carl Tropper. “A static partitioning and mapping algorithm for conservative parallel simulations”. In: *Proceedings of the eighth workshop on Parallel and distributed simulation*. 1994, pp. 164–172.

[230] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. “SWiMNet: a scalable parallel simulation testbed for wireless and mobile networks”. In: *Wireless Networks* 7.5 (2001), pp. 467–486.

[231] Azzedine Boukerche and Amber Roy. “Dynamic grid-based approach to data distribution management”. In: *Journal of Parallel and Distributed Computing* 62.3 (2002), pp. 366–392.

[232] Rodolfo Bezerra Batista, Azzedine Boukerche, and Alba Cristina Magalhaes Alves de Melo. “A parallel strategy for biological sequence alignment in restricted memory space”. In: *Journal of Parallel and Distributed Computing* 68.4 (2008), pp. 548–561.

[233] Azzedine Boukerche and Carl Tropper. “A distributed graph algorithm for the detection of local cycles and knots”. In: *IEEE Transactions on Parallel and Distributed Systems* 9.8 (1998), pp. 748–757.

[234] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. "An efficient synchronization scheme of multimedia streams in wireless and mobile systems". In: *IEEE transactions on Parallel and Distributed Systems* 13.9 (2002), pp. 911–923.

[235] Azzedine Boukerche et al. "Exploiting model independence for parallel PCS network simulation". In: *Proceedings Thirteenth Workshop on Parallel and Distributed Simulation.* 1999, pp. 166–173.

[236] Azzedine Boukerche, Amber Roy, and Neville Thomas. "Dynamic grid-based multicast group assignment in data distribution management". In: *Proceedings of the Distributed Simulation and Real-Time Applications.* 2000, pp. 47–54.

[237] Azzedine Boukerche and Caron Dzermajko. "Performance evaluation of data distribution management strategies". In: *Concurrency and Computation: Practice and Experience* 16.15 (2004), pp. 1545–1573.

[238] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. "A distributed algorithm for dynamic channel allocation". In: *Mobile Networks and Applications* 7.2 (2002), pp. 115–126.

[239] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. "An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure". In: *12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications.* 2008, pp. 61–68.

[240] Azzedine Boukerche et al. "Grid-filtered region-based data distribution management in large-scale distributed simulation systems". In: *38th Annual Simulation Symposium.* 2005, pp. 259–266.

[241] Lin Gu, Deze Zeng, and Song Guo. "Vehicular cloud computing: A survey". In: *2013 IEEE Globecom Workshops (GC Wkshps).* IEEE. 2013, pp. 403–407.

[242] Md Whaiduzzaman et al. "A survey on vehicular cloud computing". In: *Journal of Network and Computer applications* 40 (2014), pp. 325–344.

[243] DoITT NYC Gov Lab Studio. *NYC Open Data*. [Online] `https://opendata.cityofnewyork.us/`. Accessed on 2020-03-01.

[244] D. I. Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98.

[245] Yao Qin et al. "A dual-stage attention-based recurrent neural network for time series prediction". In: *arXiv preprint arXiv:1704.02971* (2017).