Freeway Travel Time Prediction Using Deep Hybrid Model – Taking Sun Yat-Sen Freeway as an Example

Pei-Ya Ting, Tomotaka Wada, *Senior Member, IEEE*, Yi-Lun Chiu, Min-Te Sun, *Member, IEEE*, Kazuya Sakai, *Member, IEEE*, Wei-Shinn Ku, *Senior Member, IEEE*, Andy An-Kai Jeng, and Jing-Shyang Hwu

***Abstract*—As the population keeps growing, traffic congestion happens more and more often. Consequently, travel time has become an important indicator of driving experience. Accurate travel time information helps drivers plan their route more wisely and thus effectively alleviate traffic congestion. In this research, we propose a vehicle travel time prediction model for freeway traffic. The data used in this research are derived from the traffic dataset of the Taiwan Freeway Bureau, and the travel time prediction is made for the Sun Yat-sen Freeway between Taipei and Hsinchu. First, the missing value of the raw data is imputed by Autoencoder. The data are then segmented according to time series and are used to build the prediction model. To effectively capture the hidden features required to predict the travel time for the vehicle traveling on the freeway, a deep learning architecture is adopted in our system, which includes the GRU neural network model, the XGBoost model, and the Hybrid model that combines the GRU and XGBoost through linear regression. To increase computational efficiency, the travel time predictions for consecutive toll gates every 5 minutes apart are pre-computed offline, so that the online travel time prediction of the whole trip can be obtained by simply summing up a few numbers. Experimental results based on actual traffic data show that the proposed system can achieve good performance in terms of prediction accuracy and execution time.**

***Index Terms*—Travel time, GRU, XGBoost, hybrid model.**

I. Introduction

TRAVEL time is an important indicator of traffic capacity and traffic efficiency. As one part of dynamic traffic information, travel time is an important concern for travelers. With real-time estimation of travel time, the Intelligent Transportation System (ITS) can provide information to decision makers for traffic control or guidance operations and to travelers for route selection. Hence, it is important to be able to provide a good estimation of the time required to traverse a specific route. Accurate travel time information helps travelers plan their routes more wisely, which effectively alleviates traffic congestion and improves operation efficiency.

In this research, we propose a travel time prediction technique based on the Gated Recurrent Unit (GRU) neural network model, which can preserve historical sequence information in its model structure for travel time prediction. The model has a deep structure in terms of time, which can be applied to travel time prediction. In addition, we also explore eXtreme Gradient Boosting (XGBoost), a tool for massively parallel boosted trees and is currently the most efficient boosted tree toolkit. By adding a new weak learner, efforts are made to correct the residuals of all the previous weak learners, and finally multiple learners are added together for final prediction to get higher prediction accuracy. Through linear regression, we combine the advantages of GRU and XGBoost models to form a new Hybrid model for travel time prediction. This Hybrid model can effectively obtain better prediction results. Unlike most travel time prediction methods that require the traffic data for each road segment to be trained separately, our approach can focus on training traffic data throughout the road network at one time. The experimental results based on real traffic data confirm that the proposed method can produce good performance in terms of prediction accuracy.

In summary, the contributions of this research are listed as follows. First, we propose a travel time prediction system. Two models, namely the GRU model and the Hybrid model, are proposed for travel time prediction. Second, in our prediction system, the travel time predictions for consecutive stations (i.e., toll gates) every 5 minutes apart are pre-computed offline, so that the travel time prediction of the whole trip can be obtained by simply summing up a few numbers. This approach significantly increases computational efficiency. Third, by using the dataset of the travel time between Taipei and Hsinchu on the Sun Yat-sen Freeway in Taiwan, we validate that our hybrid model provides the best prediction accuracy compared against other popular travel time prediction methods.

The organization of this paper is described as follows. The related work of previous research for different traffic time prediction is presented in Section II. In Section III, the design of our freeway travel time prediction system is explained in detail.

Manuscript received September 19, 2019; revised March 24, 2020; accepted May 6, 2020. Date of publication June 2, 2020; date of current version August 13, 2020. The review of this article was coordinated by Prof. J. W. Choi. *(Corresponding author: Tomotaka Wada.)*

Pei-Ya Ting is with the Taiwan Semiconductor Manufacturing Company, Hsinchu 300-78, Taiwan (e-mail: adam100830@gmail.com).

Tomotaka Wada is with the Department of Electrical and Electronic Engineering, Kansai University, Osaka 564-8680, Japan (e-mail: wadat@kansai-u.ac.jp).

Yi-Lun Chiu and Min-Te Sun are with the Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan (e-mail: yilun227@gmail.com; msun@csie.ncu.edu.tw).

Kazuya Sakai is with the Department of Electrical Engineering and Computer Science, Tokyo Metropolitan University, Hino 191-0065, Japan (e-mail: ksakai@tmu.ac.jp).

Wei-Shinn Ku is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: weishinn@auburn.edu).

Andy An-Kai Jeng and Jing-Shyang Hwu are with the Industrial Technology Research Institute, Hsinchu 310, Taiwan (e-mail: andyjeng@itri.org.tw; jshwu@itri.org.tw).

Digital Object Identifier 10.1109/TVT.2020.2999358

The hyperparameter tuning of our proposed prediction system as well as the performance results and comparison of our system and other popular methods are reported in Section IV. Finally, the conclusions of this paper and the future work are summarized in Section V.

II. Related Work

Travel-time prediction, an important issue in transportation, has been studied since the 1970 s. These research works have been focused on a model for predicting speed, volumes, and travel times on both spatial and temporal domains. With the development throughout several decades, many different approaches have been proposed to improve the prediction accuracy and efficiency. Based on the type of forecasting model, we divide these works into three types: parameter models, non-parameter models, and deep learning.

A. Parameter Models

A learning model that summarizes data with a set of parameters of fixed size which can be computed with empirical data is called a parameter model. The most widely used parameter models are Autoregressive Integrated Moving Average (ARIMA) [1], Linear Discriminant Analysis (LDA) [2], and Naïve Bayes [3]. ARIMA model is adopted in [1] to predict short-term freeway traffic data, which uses 5-minute aggregations of volume time-series data. All of the datasets are represented by an ARIMA (0,1,3) model, which varies over locations and time. Levin and Tsao [1] apply Box-Jenkins time-series analyses to predict highway traffic flow and discover that the ARIMA model is the most statistically significant one among all forecasting models. In the past, people tried to forecast the short-term traffic volume based on statistical analysis of traffic profiles. These traffic volumes are consecutive values, which are divided by the given length of time. It can be shown that the daily traffic volume profiles are grouped into different classes [4]. Kwon *et al.* [5] use more information, such as occupancy data from highway detectors and historical travel-time information, on linear discriminant analysis.

B. Non-Parameter Models

Non-parameter models refer to the models without fixed structure and parameters, which do not require making any assumption about the distribution of the population. The support vector machine (SVM) has better generalization ability for limited samples than traditional techniques. In 2004, Wu [6] applied support vector regression (SVR) for short-term prediction which can predict travel time well. In [7], Yang *et al.* propose GA-SVM model for bus arrival time prediction which has a better prediction accuracy. Since SVM has greater generalization ability, it will perform well for time series analysis, especially where traffic conditions in many countries such as India are not homogeneous and lane disciplined. In [8], Robinson *et al.* propose the method using k nearest neighbors (k-NN) technique to predict urban link travel time. In [8], it is shown that the k-NN model provides an attractive framework for aggregating travel time records from different time with similar potential traffic conditions in order to provide accurate travel time estimates. Yu *et al.* propose the model which integrated cluster analysis, principal component analysis, and k-NN method, and the model is applied to bus arrival time prediction [9]. Tak *et al.* propose a new algorithm, called multi-level k-NN [10], which is used to predict travel time with higher computational efficiency and prediction accuracy. In Taiwan, Liu proposes a new model, which uses dynamic k-NN method and the result shows that a large database can improve the accuracy of the k-NN model [11]. However, using a large amount of data in the training phase may increase the training time of a prediction model.

C. Deep Learning

In 2017, Nam *et al.* proposed a Deep Neural Networks (DNN) method [12] to predict traveler mode choice behavior. Nam *et al.* attempt to mix different structures on the DNN model to predict travel time more accurately. Also, Treethidtaphat *et al.* [13] propose a DNN model with multiple nonlinear hidden layer, which results in a model with more complex features. DNN model for bus arrival time prediction [13] at any distance of the route generally outperforms the currently used ordinary least square model in several aspects. In the same year, Yi *et al.* developed a TensorFlow DNN architecture [14] to estimate traffic conditions using real-time transportation data. The transportation data are aggregated every five minutes to train the model. Different from the previous method, Ishak *et al.* [15] propose an optimized traffic prediction method by using multiple topologies of dynamic neural networks. It is proved by experiments that no single topology is always better than the others for all the predictions. Therefore, Recurrent Neural Networks (RNN) should be a better method for travel time prediction. However, due to the structure of RNN model, older data may have little influence on the current network, and the problem of vanishing gradient problem is easy to occur. In 1997, Hochreiter *et al.* [16] proposed an new algorithm, named Long Short-Term Memory (LSTM), to improve the problem of vanishing gradient. In 2014, Cho *et al.* [17] proposed an algorithm, named Gated Recurrent Units (GRU), which can further improve the efficiency of RNN model training.

III. Design

In this section, we introduce the design of the travel time prediction system. The system is composed of five modules, including data collection, data pre-processing, prediction model design, total travel time computation, and visualization.

A. Data Collection

In order to predict vehicle travel time on a freeway, we collect data related to the travel time in addition to the historical travel time data, such as the average vehicle speed and the traffic volume. Because there are two different types of traffic volume in our dataset, we redefine them as follows.

1) *One-Station Traffic Volume:* is defined as the number of vehicles detected by each station every five minutes for each vehicle type.

2) *Two-Station Traffic Volume:* is defined as the number of vehicles detected by the upstream and downstream stations every five minutes for each vehicle type.

These data records contain the travel time of the adjacent stations every five minutes, the speed of the vehicle on different sections, and the number of vehicles passing through two toll gates. These features are arranged in chronological order as the input of the prediction model. All of the data are derived from three traffic datasets (M03 A, M04 A and M05 A) of the Taiwan Freeway Bureau.

1) *One-Station Traffic Volume (M03 A):* The M03 A dataset records the one-station traffic volume.
2) *Average Travel Time (M04 A):* The M04 A dataset records the two-station traffic volume and the average travel time every five minutes. The average travel time is the total travel time of different types of vehicles divided by the number of vehicles. For example, if there are five vehicles passing through station A and station B within five minutes, the average travel time will be the sum of the time that the five vehicles took to pass through the two stations divided by five. Therefore, if there is a vehicle getting off the interchange between two stations to the service area and then getting on the interchange again, the time the vehicle stayed in the service area will be added to the total travel time and consequently increase the average travel time. In other words, the difference between the average travel time and the actual travel time will increase. Additionally, if there is no vehicle passing through stations A and B within five minutes, the two-station traffic volume is 0 and the average travel time is also recorded as 0.
3) *Average Traffic Speed (M05 A):* The M05 A dataset records the two-station traffic volume and the average speed every five minutes. The average speed is calculated by adding the speed of each vehicle, then dividing it by the number of vehicles. Therefore, if there is no vehicle passing through stations A and B within five minutes, the two-station traffic volume is 0 and the average speed is also recorded as 0.

B. Data Pre-Processing

1) Imputation Model: In Section III-A, we point out how the M03 A, M04 A and M05 A databases are used to obtain the average travel time and average traffic speed of passing vehicles. The average travel time and average speed will be recorded as 0 if no vehicle passes through the adjacent stations within five minutes. Since most of the missing values occur in the early morning hours especially for some stations, it implies that there are often no vehicles passing through some road segments during that period of time. Because the maximum traffic speed can be known through the information of road usage, it is assumed that, in case there is no vehicle passing through during that time period, the maximum speed can be used as the imputed values of the average traffic speed. Similar to the average travel time, the imputed values of travel time can be the time which it takes for the vehicle to pass the road segments at the maximum speed. However, because both types of traffic volume cannot be imputed by the previous method, different imputation methods are needed to deal with such missing values in traffic volume.

Autoencoder is an unsupervised learning algorithm, which takes the input $X \in [0,1]^d$ and encodes it to an intermediate representation $H \in [0,1]^{d'}$, where d' represents a different dimensional subspace. The hidden layer captures the coordinates along the main factors of variation and decodes the encoded data back to the original d dimension space. The encoder and decoder of Autoencoder are both artificial neural networks, which are represented in Equation (1) and Equation (2), respectively.

$$h = s\left(Wx + b\right) \tag{1}$$

$$o = s\left(W'h + b'\right) \tag{2}$$

where o is the decoded results and s is any nonlinear function.

The Denoising AutoEncoder (DAE) [18] algorithm is used in our experiments, which is a natural extension of Autoencoder. Based on the principle of Autoencoder, DAE adds noise to the input value x, which is equivalent to applying dropout to the input layer. For example, we assume that the raw data is x, and remove some of the raw data with fixed probability distribution and randomly set it to 0, which causes the raw data losing part of the feature to become a modified dataset x'. After encoding and decoding x', we can get the recovered dataset $x'' = g(f(x'))$. The recovered data will approximate the uncontaminated raw dataset x as much as possible. We hope that the hidden layers learn robust features and are able to combat the contamination or lack of raw data to a certain extent. This method is also implemented on the MNIST dataset [19]. By adding noise to the image and training the prediction model, the final experiment proves that the features learned by DAE after adding noise are better. In the actual test data, noise is inevitable. Training the network with noise allows the network to have more generalization ability in the test data. If there is no vehicle passing through two stations within five minutes, the average travel time and average vehicle speed of the road segment will be recorded as 0 in our data, so we treat such data as missing values, and then use the principle of DAE to fill in the missing values.

The default architecture is shown in Fig. 1. We employ a typical DAE, that is, more neurons in successive hidden layers during encoding phase compared to the input layer. Mapping input data to higher dimension subspace creates representation of additional connections between consecutive layers to help with data recovery more effectively. This approach allows our DAE model to be verified empirically in supplemental materials. We start with an initial n dimension input, then add Θ neurons to each successive hidden layer, increasing the dimension to n+Θ. The k is the number of hidden layers of the encoder. We can arbitrarily select Θ according to different features, which can be adjusted as a hyperparameter to meet the expected results. Our model inputs are standardized between 0 and 1 to facilitate faster convergence for small to moderate sample sizes. In the decoding phase, the opposite of the encoding phase is used, which means the number of neurons is reduced as the data approach the output layer by layer.

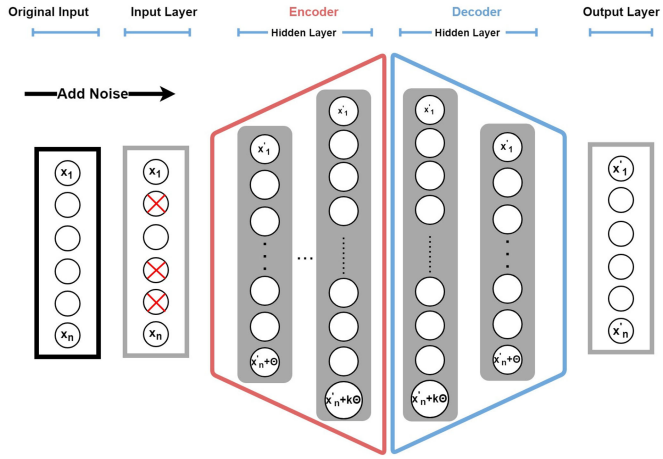

Fig. 1. Denoising AutoEncoder architecture.

2) Feature Scaling: In neural networks, feature scaling can speed up gradient descent and improve model accuracy. For example, one of the features we use in our research is average travel time, and this feature may vary greatly for different road segments or time periods. If the variation of a feature is too large, the cost function of the model may be dominated by this feature. The feature scaling used in our research is Min-Max normalization (i.e., Rescaling), and the formula is presented in Equation (3). Min-Max normalization brings all values to the range of 0 and 1. Compared with the standardization, Min-Max normalization can reduce the impact of singular value [20]. Note that the features selected in this research are average travel time, average speed, and two types of traffic volume. These features are likely to have singular values as traffic conditions can vary from time to time. For example, an increase in travel time can be caused by traffic jams making the travel time a lot longer. However, such a situation is not unreasonable, so we will treat it as the singular value, and keep it instead of removing it. Feature scaling helps to reduce the effects of singular values.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3}$$

3) Time-Level Sliding Window Sequence: According to the characteristics of the time series model, our model will be trained by the training dataset according to the time series to obtain the final model. In Section III-A, the traffic data (e.g., average speed, average travel time and two types of traffic volume) are sorted by time. If the data are fed into the model without pre-processing, the model can only use the previous five minutes of traffic information to predict travel time for the next time segment. However, highways usually have traffic jams in rush hours, so we try to use a longer time period to predict traffic information at the next time period. Before training the model, we will split the dataset into new data types, a process called time-level sliding window sequence generation. As shown in Fig. 2, the length of each input data is set as the length of sliding window, which is denoted as N. The first input uses the data from time t to $t+4$ as the feature, and the data at time $t+5$ as its label. The second input will use the data from $t+1$ to

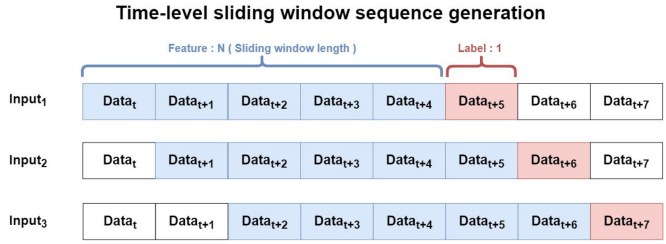

Fig. 2. Time-level sliding window sequence.

$t+5$ as the feature, which is the first input moving forward by one time slot. The data at time $t+6$ will be used as the label. Similarly, the third input uses the data from time $t+2$ to $t+6$ as the feature and the data at time $t+7$ as the label.

C. Prediction Model Design

After the data pre-processing, the next step is to construct the prediction model. We propose three methods in this research. The first two use the GRU model for travel time prediction. The third one uses the hybrid model in which both the GRU and XGBoost models are adopted and the results from both models are integrated to predict travel time. In the following sections, the prediction models will be described in detail. The section includes the model architecture, whether to use normalization, and the parameter selection of time-level sliding window sequence. Note that different pre-processing will be applied for different models.

1) GRU Model: GRU models are commonly used for a time series problem. The average speed, average travel time, one-station traffic volume as well as the two-station traffic volume are the features in the input layer. Since the correlation between these four features is high, we can combine these features as the input so that the model can explore the correlation between the features. To achieve this, four different features are separately normalized in the pre-processing stage, and the data are processed by the time-level sliding window sequence. Feature scaling allows the model to optimize the gradient descent and improve the prediction accuracy. The sliding window allows the model to learn the temporal and spatial correlation between stations and between routes. Finally, all features are concatenated into a dataset and used as the input to the GRU model. By doing so, the model is able to learn not only the temporal and spatial correlations between different stations and routes but also the relationship between different features of the same route. This can effectively increase the accuracy of model prediction and increase the speed of model training. In addition, in order to prevent over-fitting, early stopping is used in the model training. This mechanism is to observe the validation loss during training. If the loss does not decrease for 10 epochs compared to the lowest loss of the previous epoch, the training is stopped. The architecture is shown in Fig. 3.

2) Hybrid Model: Travel time prediction is a continuous target variable autoregressive prediction problem. Many models can solve such problems. Therefore, we try to combine the XGBoost and GRU models. XGBoost is a gradient boosting

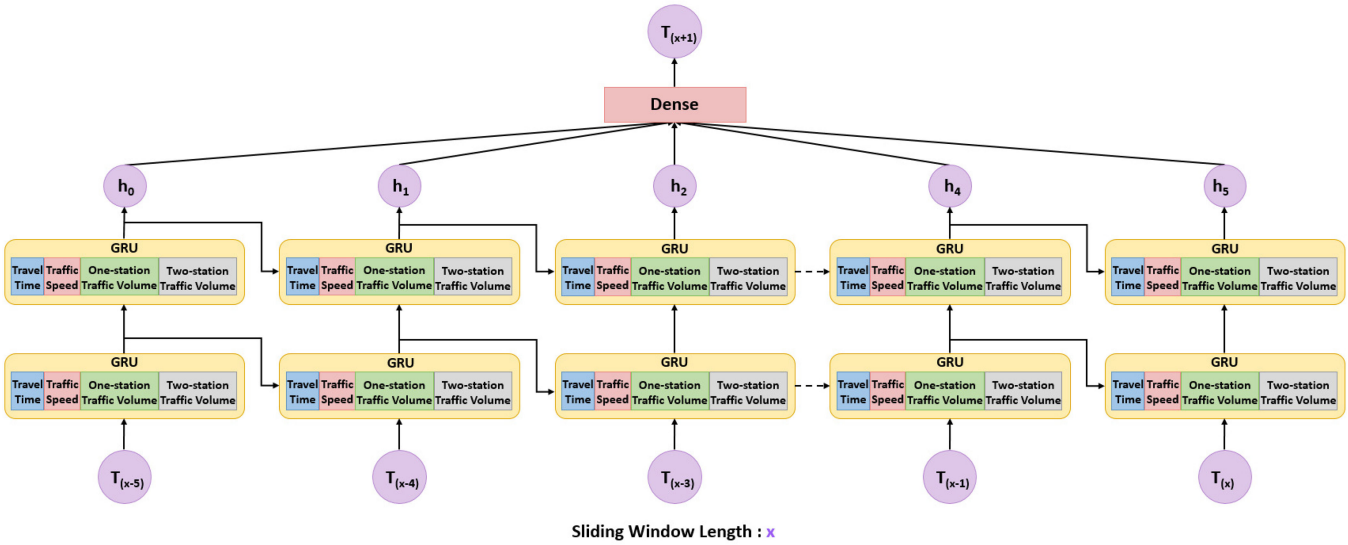

Fig. 3. The architecture of the second GRU models.

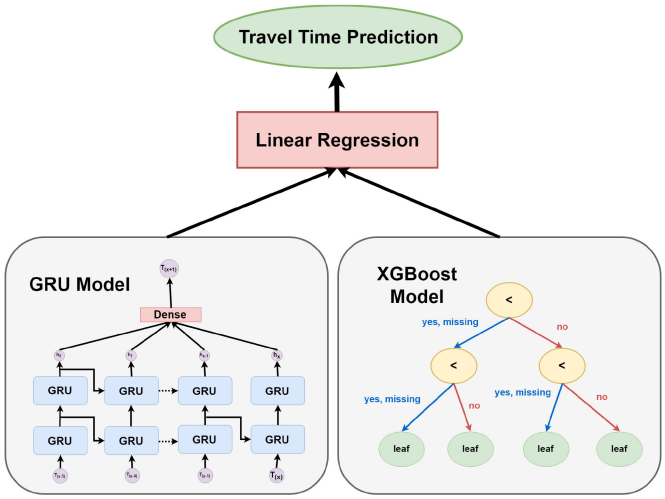

Fig. 4. The architecture of Hybrid models.

decision tree that can be used for classification or regression, but is not a method for dealing with typical time series problems. Conversely, GRU is a neural network model, which is a derivative of the RNN model and is designed to deal with time series problems. The principle of the two models is quite different, and the correlation of the results is small. Therefore, the fusion of two different models is conducive to improving the accuracy of prediction. The architecture of the hybrid model is shown in Fig. 4.

The raw data of travel time are used as a model input. Since XGBoost is different from the general gradient boosting decision tree, it incorporates the functionality of Regularization and can effectively solve the problem of over-fitting. This effectively controls the complexity of the model, so there is no need to normalize the features. Using XGBoost makes the model simpler. In addition to effectively preventing over-fitting, the training time is significantly reduced. That is why XGBoost is superior to the traditional gradient boosting decision tree. As for data pre-processing, the time-level sliding window sequence is conducted for data and the sliding window length is set as 2. It means that these models will use the travel time of the previous 10 minutes to predict the travel time of the next five minutes. The model uses grid search to find the best parameters of XGBoost. Finally, the results predicted by the final GRU model and the XGBoost model will be used as inputs to the linear regression model, and the final results will be obtained through the regression model. The linear regression model is chosen here because the total travel time and many selected features, including the reciprocal of speed (i.e., the amount of time spent for each unit travel distance), have a linear relationship.

D. Computation of Total Travel Time Prediction

In the travel time prediction system, we need to calculate the travel time prediction of the whole trip. Our model only predicts the travel time from one station to the next. As a result, to obtain the total travel time for the whole trip, which may include several stations, the travel time predictions for all pairs of consecutive stations need to be added together. In addition, for fast computation of the prediction of the whole travel time, the prediction of the travel time for each pair of consecutive stations is actually pre-computed every 5 minutes apart. To ensure the predicted travel time for a pair of consecutive stations at a given moment to be as accurate as possible, the travel time prediction of that pair closest to the moment is adopted. As an example, let us assume the prediction has been pre-computed for every pair of consecutive stations at 00, 05, 10, 15, 20, 25, 30, 35, 40, 45, and 55 minute of every hour. Fig. 5 shows that a vehicle reaches $Station_2$ at 08:00. Let the pre-computed travel time prediction at 8:00 for vehicles traveling from $Station_2$ to $Station_3$ ($Route_B$) to be 12 minutes. For the travel time from $Station_3$ to $Station_4$ ($Route_C$), we will adopt the pre-computed prediction for 08:10, as 08:10 is closer to 08:12 (the predicted arrival time at $Station_3$ for the vehicle) than 08:15.

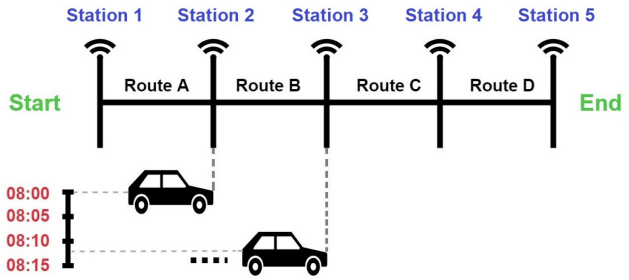

Fig. 5. An example of the total travel time computation.

Fig. 6. The travel time prediction website.

E. Visualization

In this research, the system realizes the result of highway travel time prediction through an interactive website. The interactive website is implemented by Shiny [21], which is a package of R language [22] to implement the front-end and back-end of the website and provide a graphical interface so that the chart can be rendered dynamically on the website. The basic Shiny project consists of two parts, which are elaborated as follows.

1) *ui.R:* The front-end program is responsible for describing how the project page should be rendered and typeset. In our system, the display of Google Maps [23] and travel itineraries will be shown here, and the predicted travel time will be provided to the user.
2) *server.R:* The back-end program is responsible for data analysis, program operations, and visualization. Finally, the results will pass to the front-end. Our system will use the back-end program to predict travel time, and the estimated time spent by the user will be summed up in the post-processing stage.

The system completes the interactive website through the Shiny package and places the website on shinyapps.io. [24]. The user can select a specific time from the interactive website and view the travel time required to travel the highway at that point in time. The traffic condition of each road segment will be distinguished by color, which is shown in Fig. 6. At the same time, users can also predict the travel time required to travel from a starting point to a destination on the highway through the website. If a user wants to know the time required to travel a specific road segment for the previous five minutes or the next five minutes, she can move the cursor to the road segment on Google Maps and get the information through the label.

IV. Performance

In this research, the performance of deep learning-based travel time prediction algorithms, including GRU, XGBoost, and Hybrid models, is evaluated. In the following subsections, the data description is first presented. To assess the performance of the travel time prediction system, Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) are adopted. A Windows desktop equipped with Intel Core i7-6700 CPU, RTX 2060, and 32 GB RAM is used for model training.

TABLE I
Traffic Data Description

	M03A	M04A	M05A
Time Period	2018/07/01 2018/12/31	2018/07/01 2018/12/31	2018/07/01 2018/12/31
Stations/Routes	61	69	69
Data Entries	52992	52992	52992

A. Data Description

The datasets used in this research include traffic volume, average travel time, average traffic speed, and the source data of each travel route. The traffic data are collected every five minutes from 61 individual detectors, which are deployed in freeway systems from Taipei to Hsinchu. There are 69 road segments between Taipei and Hsinchu.

The collected data are aggregated every 5 minutes for each detector station. In our experiment, we select the travel time of the past hour, which actually is the time sequence of 12 data points, to predict the travel time in the next 5 minutes. The traffic data are collected from July 1, 2018 to December 31, 2018. The data are divided into two parts. The first four months (i.e. the data from July, 2018 to October, 2018) are used to train our model. The last two months (i.e. the data from November, 2018 to December, 2018) are used to evaluate the model.

In our research, the missing data account for 1% of the raw data. Although the missing data only occupy a small part of the dataset, we still train an AutoEncoder model to impute the missing data. After data imputation, the traffic dataset can effectively improve the accuracy of the prediction. The evaluation of imputation model will be explained in Section IV-B2. The information about the traffic data is shown in Table I. The traffic information for different types of vehicles can be obtained from the traffic datasets. However, the most commonly used type is the general vehicle (i.e., car), and the general vehicle has the most complete traffic information. Therefore, the purpose of this research is to predict the travel time of the general vehicle.

B. Evaluation

In this section, the most common performance metrics to measure accuracy for continuous variables are introduced, which are Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE). Since the first two are relative error metrics, they are not affected by the unit or the size of the value. The last one is an absolute error metric that can be used to determine the degree of difference between the predicted value and the actual value. These performance metrics are explained as follows.

1) Performance Metrics:

i) *MAPE:* Mean Absolute Percentage Error is used as an evaluation criterion for the prediction models. Because MAPE is a relative value, it is not affected by the unit and size of the predicted value. The degree of difference between the predicted value and the actual value can be objectively obtained. Therefore, it is applied to the research to obtain the accuracy of the prediction results, which is shown in Equation (4).

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y_i}}{y_i}\right| \times 100\% \tag{4}$$

The closer the MAPE value is to 0, the better the prediction result is.

ii) *RMSE:* The Root Mean Square Error represents the square root of the average of the square of the differences between predicted values and observed values. It can assess the reliability of model predictions. If the root mean square error is small, it indicates that the prediction model is reliable. The computation of RMSE is shown in Equation (5).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\hat{y_i} - y_i\right)^2}{N}} \tag{5}$$

iii) *MAE:* Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. The MAE can avoid the problem that the positive and negative errors cancel out each other. Therefore, the error of the prediction can be accurately reflected. Its computation is shown in Equation (6).

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}| \tag{6}$$

In order to evaluate the predicted results in different aspects, all three aforementioned performance metrics, including the accuracy of the prediction (MAPE), the reliability (RMSE), and the error range (MAE), are used as the evaluation indicators of this research.

2) Experimental Results:

i) *Data Imputation:* All traffic information in this research comes from the Freeway Bureau of Taiwan [25]. As mentioned in Section III-A, the average travel time and the average speed of missing values are recorded as 0, because there is no vehicle passing through during that time period. However, these values are not adversarial to model training, so we consider such values as missing values. The missing values caused by the absence of a vehicle is approximately 0.2% in our dataset. Therefore, the data imputation will not cause too much distortion of the raw data, and can effectively improve the prediction accuracy of the model. In model tuning, we select neurons, hidden layers, and learning rate as the hyperparameters to be tuned, and find that the number of neurons has the biggest impact on the accuracy of the model. We have different models for the four features of missing data to implement data imputation. As shown in Table II, different neurons are used for different datasets, and the table shows the best prediction results. Compared with the one-station traffic volume and two-station traffic volume, the changes of the average travel time and the average travel speed are relatively stable. Therefore, when training the former datasets, more neurons are needed to achieve the higher accuracy of the model.

TABLE II
THE PERFORMANCE OF MISSING DATA IMPUTATION

	Travel Time	**Traffic Speed**	**Two-station Traffic Volume**	**One-station Traffic Volume**
Neurons	[512,512,512]	[512,512,512]	[1024,1024,1024]	[512,1024,2048]
Acc[1]	91.00%	93.56%	88.30%	88.06%
MAE	21.80	6.00	10.20	20.66
RMSE	68.80	10.08	18.11	30.99

[1]Accuracy (i.e., Acc) is calculated as (1 - MAPE) * 100%

TABLE III
PERFORMANCE ON DIFFERENT METHODS OF DATA IMPUTATION

Models	RMSE		Accuracy		MAE	
	Solution 1	Solution 2	Solution 1	Solution 2	Solution 1	Solution 2
GRU	72.56	72.62	93.66	93.84	15.09	14.66
XGBoost	70.70	70.67	94.100	94.102	14.02	14.02
Hybrid model	70.89	70.25	94.16	94.21	13.89	13.77
k-NN	84.61	84.67	90.34	90.76	22.97	21.98

TABLE IV
THE MAE OF GRU MODEL

Length of sliding window	4 Layers		5 Layers		6 Layers	
	MAE	MSE	MAE	MSE	MAE	MSE
6	**14.66**	17.42	14.72	18.32	14.73	17.66
12	76.82	17.85	15.05	17.42	15.15	18.06
24	76.82	18.43	15.50	17.98	15.16	17.80

There are two methods of data imputation mentioned in Section III-B1. One is to use the maximum speed for the missing values of the average traffic speed, and the time required to pass the road segment with the maximum traffic speed is used as the missing values of the average travel time. We use two solutions. The first solution uses both maximum speed and DAE as the imputation methods. The second solution uses DAE to impute the missing value for all datasets. We apply two different solutions to the generated models, and we can find that the prediction results are similar for both models. The evaluation of model tuning will be explained in the next paragraph.

ii) *Model Tuning:* The M04 A dataset contains a travel time history dataset of 69 routes. This research uses GRU and XGBoost models to predict travel time and improve prediction accuracy by using the hybrid models of GRU and XGBoost models. The following describes the hyperparameter selection of GRU and XGBoost. The two prediction methods are compared with raw data, and RMSE, MAE, and accuracy are used as performance metrics. Since tuning the model with different numbers of neurons does not significantly increase the accuracy of the GRU model, we choose number of hidden layers, loss function, and the length of sliding window as the hyperparameters to be tuned. As shown in Table IV, when the length of sliding window is longer, more hidden layers

TABLE V
THE TRAINING TIME OF GRU MODEL

Length of sliding window	4 Layers		5 Layers		6 Layers	
	MAE	MSE	MAE	MSE	MAE	MSE
6	460	**320**	910	424	998	507
12	189	690	1207	695	1069	877
24	210	1168	1367	1250	2219	1601

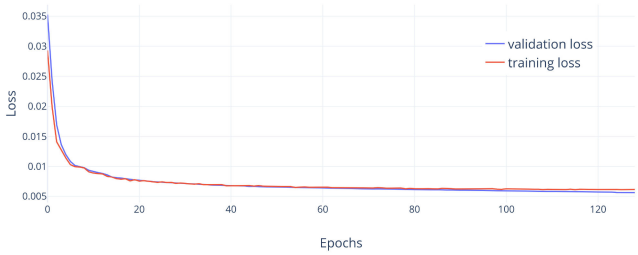

Fig. 7. Learning curves of GRU model.

TABLE VI
THE MAE OF XGBOOST MODEL

Length of sliding window	Max depth 4		Max depth 5		Max depth 6	
	0.05	0.1	0.05	0.1	0.05	0.1
2	14.37	14.28	14.24	14.21	14.22	14.23
6	14.20	14.09	14.07	14.03	**14.02**	14.04
12	14.25	14.17	14.10	14.12	14.06	14.14

[1]0.05 and 0.1 are learning rates

TABLE VII
THE TRAINING TIME OF XGBOOST MODEL

Length of sliding window	Max depth 4		Max depth 5		Max depth 6	
	0.05	0.1	0.05	0.1	0.05	0.1
2	**156**	157	205	185	215	220
6	406	416	499	497	570	577
12	816	856	1061	957	1127	1104

[1]0.05 and 0.1 are learning rates

are needed to improve the accuracy of the prediction model. However, when the length of sliding window is set to 6 (use the previous half hour of traffic information to predict travel time for the next time segment), using 4 hidden layers can get better results than using 6 hidden layers, which requires less training time and has a higher accuracy. Therefore, using fewer hidden layers takes less time and can effectively reduce the training costs.

A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values. The learning curves of GRU model are shown in Fig. 7. The training loss and the validation loss decrease to a point of stability, and the validation loss has a small gap with the training loss. Since continuing training of a good fit will likely lead to model overfitting, the early stopping mechanism is adopted to reduce the probability of overfitting when training the model. In model tuning, the learning rate, max depth, and the length of sliding window are chosen as the hyperparameters. As shown in Table VI, the deeper the XGBoost tree, the better the prediction can be. However, overfitting may occur when the depth of the XGBoost tree is set too deep. In order to avoid overfitting, we set the max depth to be between 4 and 6, and adjust it via cross validation. When the length of sliding window is set to 6 (use the previous half hour of traffic information to predict travel time for the next time segment), the accuracy of the prediction model cannot be improved regardless of the values of the other hyperparameters. Therefore, using a shorter sliding window can improve accuracy and effectively reduce training time and training costs.

TABLE VIII
THE COMPARISON OF PERFORMANCE METRICS

	k-NN	GRU	XGBoost	Hybrid model
RMSE	84.67	72.62	70.67	**70.25**
MAE	21.98	14.66	14.02	**13.77**
Acc	90.76	93.84	94.10	**94.21**
Training model Time (s)	**4**	460	570	1283
Execution time (s)	5184	30	**16**	47

iii) *Travel Time Prediction:* Our dataset is divided into two parts, training dataset and test dataset, which is used to evaluate the performance of the four models. This research compares the prediction accuracy with the k-NN model proposed by Liu in 2017 [11], because the experimental areas are similar, and the k-NN is also one of the methods commonly used in travel time prediction in the recent years. Table VIII summarizes the prediction results using the GRU model, XGBoost model, hybrid model, and k-NN model. As shown in Table VIII, the RMSE and MAE of GRU, XGBoost, and the Hybrid model are smaller than those of k-NN model. GRU model and XGBoost model can effectively improve the prediction accuracy. The hybrid model has a higher accuracy than the GRU model and the XBG model. Moreover, the Hybrid model can also reduce prediction errors. Different from other models, k-NN is quite fast when training the model; however, it takes a lot of time for prediction, especially when the dataset is large. Since the training time and the predicted execution time are quite different, we compare the model training time and the predicted execution time separately in the following.

The Empirical Cumulative Distribution Function (ECDF) is regarded as the value of a measured variable whose distribution function value represents the proportion of samples in all observed samples that are less than or equal to the value. We compare the travel time predicted by different models with the raw data, and convert the MAE into ECDF. As shown in Fig. 8, the hybrid model, GRU model, and XGBoost model occupy more proportions than the k-NN model within 50 seconds of the MAE. Most of the prediction errors are within 50 seconds. When the MAE is 250, the percentages of the error of the four prediction models are similar, because most of the data with MAE higher than 250 come from the road segments with service area.

As discussed previously, there are two road segments that will go past the service area. To prevent the influence of

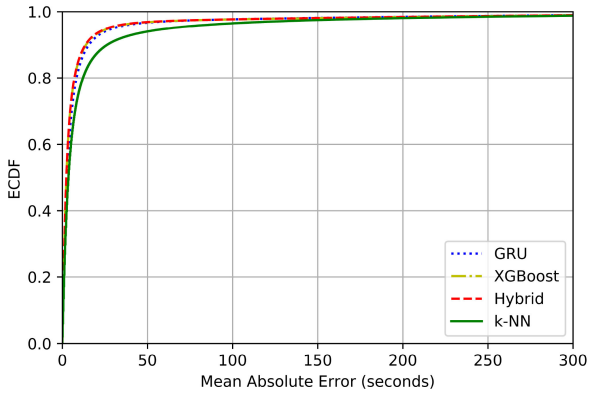

Fig. 8. The empirical cumulative distribution function of 4 models.

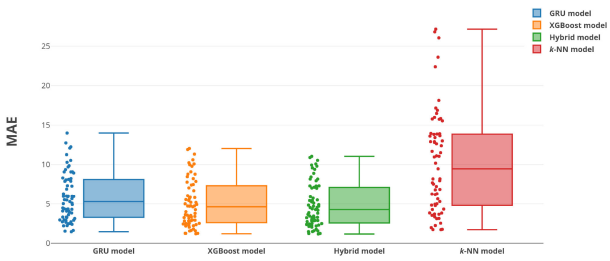

Fig. 9. The prediction error of each stations indicate by points. The error distribution indicated by box plots.

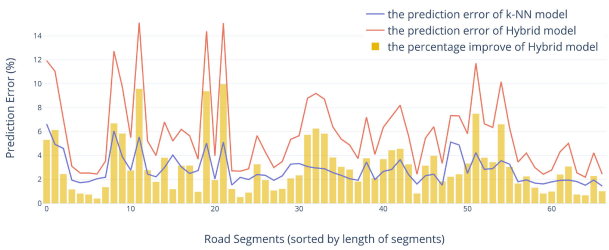

Fig. 10. The prediction error of Hybrid model and k-NN model indicate by line chart. The error improvement indicated by bar chart.

staying time in the service area, these two road segments are excluded in the following discussion. As shown in Fig. 9, the points are used to represent the prediction errors of 67 stations, and the box plots are used to indicate the error distribution of the station. The k-NN model has large prediction errors, which are widely distributed in areas with higher MAE. Especially in some stations, the prediction error is particularly large. Our models have low prediction errors. However, the prediction accuracy of GRU model in some stations is not good. The Hybrid model obtained by combining GRU model and XGBoost model can concentrate the distribution of prediction error in the area with smaller MAE.

Considering the difference in length of each route, the percentage of prediction error of Hybrid model and k-NN at each station can be observed in Fig. 10. As shown in the line graph of the figure, Hybrid model constantly has a prediction error lower than 8% at all stations. On the other hand, the prediction error of the k-NN model exceeds 10% at some stations. The bar chart of the figure is used to indicate the improvement of Hybrid model, and the improvement is as much as 10% at some stations.

V. Conclusion and Future Work

Travel time is an important indicator of traffic conditions. In this research, we propose a travel time prediction system for freeway travel. This prediction system includes data collection, data pre-processing, prediction model design, total travel time computation, and visualization. We explore two deep learning models, the GRU model and XGBoost model, and apply them to travel time prediction. The optimal structure of the model is obtained through hyperparameter selection. By pre-processing the data through the time-level sliding window sequence, the training data contain more historical features, which enhances the prediction effect of the time series GRU model and XGBoost. Finally, the GRU and XGBoost models are combined through linear regression and the best model is obtained. We compare the performance of our models with that of the k-NN model proposed by Liu in 2017, as the experimental settings of Liu's model are similar to ours. Evaluation results show that our GRU, XGBoost, and Hybrid models can obtain higher prediction accuracy than k-NN model, and the Hybrid model obtains the highest prediction accuracy. When conducting model prediction, the execution time required by k-NN model takes as much as 10 times longer than what is required for the Hybrid model. The travel time predictions for consecutive stations every 5 minutes apart are pre-computed offline into a table, so that the online total travel time prediction can be obtained by simply summing up a few numbers in the table. This approach is especially suitable for embedded systems with limited computing resources. Finally, the prediction of travel time can effectively alleviate traffic congestion and improve driving efficiency. Last, the experiments show that the Hybrid model has the highest accuracy.

For future work, we would like to acquire more detailed traffic information possibly from the open data released by the government, and improve the prediction model by removing the outliers of travel time. Furthermore, with the rapid development of deep learning, we would like to try other different types of models (e.g., dense net) or combine these models to get more efficient and accurate predictions.

References

[1] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using Box-Jenkins techniques," *Transp. Res. Rec.*, no. 722, pp. 1–9, 1979. [Online]. Available: https://trid.trb.org/view/148123

[2] J. Rice and E. Van Zwet, "A simple and effective method for predicting travel times on freeways," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 3, pp. 200–207, Sep. 2004.

[3] X. Fei, C.-C. Lu, and K. Liu, "A Bayesian dynamic linear model approach for real-time short-term freeway travel time prediction," *Transp. Res. Part C: Emerg. Technologies*, vol. 19, no. 6, pp. 1306–1318, 2011.

[4] M. Danech-Pajouh and M. Aron, "ATHENA: A method for short-term inter-urban motorway traffic forecasting," *Recherche Transports Sécurité*, no. 6, 1991. [Online]. Available: https://trid.trb.org/view/376928

[5] J. Kwon, B. Coifman, and P. Bickel, "Day-to-day travel-time trends and travel-time prediction from loop-detector data," *Transp. Res. Record.*, vol. 1717, no. 1, pp. 120–129, 2000.

[6] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, Dec. 2004.

[7] M. Yang, C. Chen, L. Wang, X. Yan, and L. Zhou, "Bus arrival time prediction using support vector machine with genetic algorithm," *Neural Netw. World*, vol. 26, no. 3, pp. 205–217, 2016.
[8] S. Robinson and J. Polak, "Modeling urban link travel time with inductive loop detector data by using the *k*-NN method," *Transp. Res. Record: J. Transp. Res. Board*, vol. 1935, pp. 47–56, 2005.
[9] B. Yu, W. H. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transp. Res. Part C: Emerg. Technologies*, vol. 19, no. 6, pp. 1157–1170, 2011.
[10] S. Tak, S. Kim, K. Jang, and H. Yeo, "Real-time travel time prediction using multi-level k-nearest neighbor algorithm and data fusion method," in *Proc. Comput. Civil Building Eng.*, 2014, pp. 1861–1868.
[11] H.-N. Liu, "The study of travel time prediction for freeway by using dynamic *k*-NN method," Master Thesis, Nat. Chiao Tung Univ., Taiwan, pp. 1–32, 2016. [Online]. Available: https://www.airitilibrary.com/Publication/alDetailedMesh?docid=U0030-0803201714321568
[12] D. Nam, H. Kim, J. Cho, and R. Jayakrishnan, "A model based on deep learning for predicting travel mode choice," in *Proc. Transp. Res. Board 96th Annu. Meeting Transp. Res. Board*, 2017, pp. 8–12.
[13] W. Treethidtaphat, W. Pattara-Atikom, and S. Khaimook, "Bus arrival time prediction at any distance of bus route using deep neural network model," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 988–992.
[14] H. Yi, H. Jung, and S. Bae, "Deep neural networks for traffic flow prediction," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, 2017, pp. 328–331.
[15] S. Ishak, P. Kotha, and C. Alecsandru, "Optimization of dynamic neural network performance for short-term traffic prediction," *Transp. Res. Record.*, vol. 1836, no. 1, pp. 45–56, 2003.
[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[17] K. Cho *et al.*, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. SSST-8, Eighth Workshop Syntax, Semantics Struct. Statistical Trans.*, 2014, pp. 103–111.
[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.
[19] "Denoising autoencoder implemented on the mnist dataset," Mar. 15, 2015. [Online]. Available: http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model/, Accessed on: Jun. 1, 2020.
[20] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn," in *Proc. ICML Workshop AutoML*, 2014, pp. 2825–2830.
[21] C. Beeley, *Web Application Development with R Using Shiny*. Birmingham, U.K.: Packt Publishing Ltd., 2013.
[22] "The comprehensive R archive network," May 10, 2012. [Online]. Available: https://cran.r-project.org/, Accessed on: Jun. 1, 2020.
[23] "Google map platform," [Online]. Available: https://cloud.google.com/maps-platform/, Accessed on: Jun. 1, 2020.
[24] "shinyapps.io by rstudio," [Online]. Available: https://www.shinyapps.io/, Accessed on: Jun. 1, 2020.
[25] F. Bureau, "Freeway bureau,motc," [Online]. Available: https://www.freeway.gov.tw/english/Default.aspx/, Accessed on: Jun. 1, 2020.

Pei-Ya Ting received the B.S. degree in computer science from the National Taichung University of Education in 2017, and the M.S. degree in computer science from National Central University, in 2019.

Tomotaka Wada (Senior Member, IEEE) received the Ph.D. degree in communications engineering from Osaka University, Japan, in 1999. Since 1999, he was with the Faculty of Systems Engineering, Wakayama University, as a Research Associate. In 2006, he joined the Faculty of Engineering, Kansai University, where he is currently an Associate Professor. His research interests are in mobile communications, ad-hoc networks, and intelligent transport systems (ITS).

Yi-Lun Chiu received the B.S. degree in computer science from National Central University, in 2019. He is currently working toward the master's degree at the Department of Computer Science and Information Engineering, National Central University, Taiwan.

Min-Te Sun (Member, IEEE) received the B.S. degree in mathematics from National Taiwan University, in 1991, the M.S. degree in computer science from Indiana University, in 1995, and the Ph.D. degree in computer and information science from the Ohio State University, in 2002. Since 2008, he has been with the Department of Computer Science and Information Engineering at National Central University, Taiwan. His research interests include distributed algorithm design and wireless network protocol development.

Kazuya Sakai (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Ohio State University, in 2013. He is currently an Associate Professor at the Department of Electrical Engineering and Computer Science, Tokyo Metropolitan University. His research interests are in the area of wireless and mobile computing, information and network security, and distributed algorithms. He received the IEEE Computer Society Japan Chapter Young Author Award 2016. He is a member of ACM.

Wei-Shinn Ku (Senior Member, IEEE) received the M.S. degree in computer science and the M.S. degree in electrical engineering, in 2003 and 2006, respectively, and the Ph.D. degree in computer science, in 2007, from the University of Southern California (USC). He is a Program Director at the National Science Foundation and a Professor with the Department of Computer Science and Software Engineering at Auburn University. His research interests include databases, data science, mobile computing, and cybersecurity. He has published more than 120 research papers in refereed international journals and conference proceedings. He is a member of the ACM SIGSPATIAL.

Andy An-Kai Jeng received the Ph.D. degree in computer science from Nation Chiao Tung University, Taiwan, in 2007. He was a Postdoctoral Fellow at NationChiao Tung University from 2007 to 2011. He is now the Deputy Division Director of Connected and Autonomous Vehicle Division in Industrial Technology Research Institute (ITRI), Taiwan. His research interests include autonomous driving vehicle and connected vehicle technology. He received SAE the Excellent Young Engineer Awards in 2017, Edison Awards in 2018, and ITS World Congress Industry Awards in 2019.

Jing-Shyang Hwu received the Ph.D. degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2005. He is currently the Manager of the Connected Transportation Application Department, Connected and Autonomous Vehicle Division in Industrial Technology Research Institute (ITRI), Taiwan. His research interests include intelligent transportation system and connected vehicle technology.