TRR
JOURNAL OF THE TRANSPORTATION RESEARCH BOARD

Emerging Technologies, Data, and Information

Transportation Research Record
2019, Vol. 2673(4) 145–157
© National Academy of Sciences:
Transportation Research Board 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0361198119838508
journals.sagepub.com/home/trr
SAGE

Deep Learning System for Travel Speed Predictions on Multiple Arterial Road Segments

Hoang Nguyen[1], Christopher Bentley[1], Le Minh Kieu[1], Yushuai Fu[1], and Chen Cai[1]

Abstract
Accurate travel speed prediction is a critical tool for incidence response management. The complex dynamics of transport systems render model-based prediction extremely challenging. However, the large amounts of available vehicle speed data contain the complex interdependencies of the target travel speed; the data itself can be used to generate accurate predictions using deep learning methods. In this work, a deep learning methodology involving feature generation, model development, and model deployment is presented. The authors demonstrate the high performance of deep learning methods (relative to more traditional benchmarks) in predicting travel speeds from 5–30 min in advance, for a challenging arterial road network. In this study, different deep learning architectures that exploit both spatial and temporal information for several time frames are compared and analyzed. Finally, the authors demonstrate the integration of their deep learning method into a visualization system that can be directly applied for vehicle speed prediction in real time. The model-selection analysis and data-to-visualization framework in this manuscript provide a step towards decision support for incident management; for practical implementation, the predictive power of deep learning models under incident conditions should continue to be investigated and improved.

At Transport Management Centers (TMCs), the operators have to monitor traffic conditions and process incidents in a very large metropolitan area. Their main tasks include continuous observation of traffic conditions using CCTV to identify any delay or accident and then decide appropriate plans to clear the incidents on time. A real-time traffic-condition monitoring and prediction system is expected to generate early warnings to operators for abnormal traffic patterns, to facilitate better planning to respond to incidents and reduce congestion. The traffic condition on a road segment is usually evaluated using the current travel speed. The New South Wales (NSW) TMC and the research organization Data61 in Sydney have collaborated to develop a system to efficiently predict travel speeds for multiple road segments in an area. Many previous methods and systems for travel speed/travel time predictions mainly focused on single-road motorway traffic data without the need to consider traffic on connected or nearby roads (*1–6*). In this paper, a system based on deep neural networks to predict and visualize vehicle speeds on multiple arterial road segments is presented. This system is able to consider traffic conditions on connected or related road segments and forecast travel speeds for multiple roads in the area from 5–30 min ahead. Furthermore, this approach is not limited to freeway roads but is applicable to any arterial road networks where travel speed data is available. Note that the predictive models incorporated into this system are effective under regular traffic conditions, as evaluated in this work. As the models are evaluated and extended for higher performance under incident conditions, these extensions can also be integrated into the prediction and visualization system presented here.

In this work, the literature related to the proposed model is first reviewed. The deep learning methodology is then introduced, which includes the preparation of the data, the elements of the deep learning model, and model integration approach for real traffic systems. Subsequently, vehicle prediction results for deep learning and traditional vehicle speed prediction approaches are

[1]Data61 – Commonwealth Scientific and Industrial Research Organisation (CSIRO), Eveleigh, NSW, Australia

Corresponding Author:
Address correspondence to Hoang Nguyen: hoang.nguyen@data61.csiro.au

presented. Finally, the effective application of the proposed prediction scheme is illustrated using a visualization system and its implementation in real-world traffic management is discussed. This implementation could be directly applied to identify abnormal traffic patterns by examining the difference between predictions (trained under regular conditions) and the actual state of the traffic.

Related Work

Short-term travel speed prediction has been an active topic in the literature; for reviews of data-driven approaches see Oh *et al.* and Vlahogianni *et al.* (*6*, *7*). The existing approaches can be divided into parametric and non-parametric models.

Parametric models in short-term travel speed prediction aim to find a parametric formulation between known variables such as the previous vehicle speed on a segment, and the target variable such as the predicted speed on the given segment. The ARIMA model (*8*) is a linear function of past observations and random error terms, which has been widely applied because of its simplicity and good performance in forecasting linear and stationary time series. Seasonality variables have also been included in an extension of the ARIMA model (SARIMA) (*9*), where the authors forecast the travel time using the bluetooth technology on arterials in Brisbane. However, both ARIMA and SARIMA assume linear relationships between time-lagged variables, and they may thus fail to capture the complexity of the intrinsically nonlinear travel speed dependencies. Apart from the popular ARIMA-based models, other parametric approaches to short-term travel speed prediction include linear regression and ridge regression (*10*, *11*). Although the parametric approaches are highly scalable and transferable, they often require an assumption of the distribution of data, for example, the normal distribution in ARIMA.

Non-parametric models have been a popular emerging approach for short-term travel speed prediction. Unlike parametric models, non-parametric models do not require any assumptions about the distribution of data. There are several methods, such as Gaussian process, k-nearest neighbor and support vector regression (*1*, *12*, *13*). Neural-network-based methods are also applied widely in forecasting travel speed or travel time (*14–17*).

Deep learning is a recent non-parametric neural-network-based approach to short-term speed prediction, which attempts to model the complex nature of traffic data. Traffic flow prediction has been performed using a variety of deep learning architectures (see Ali and Mahmood (*18*) for a review). The variety of architectures applied, such as autoencoders and deep belief networks (*19–21*), reflects the different information that can be included in the input data, as well as different approaches to help the artificial neural network learn the desired mapping from input to output.

Predictive networks exploit temporal information in some form, since the traffic volume or speed at a given time is strongly correlated with that just a few minutes later, and longer periodic trends (e.g., peak-hour behavior) are also evident in traffic data. For instance, the data can be pre-processed to include temporal information as input to a simple feed-forward neural network, as in Polson and Sokolov (*22*). Alternately, recurrent neural networks (RNNs) can learn to process the time-sequence data themselves. RNNs were designed to learn temporal correlations from sequences of data and have been applied to traffic flow prediction in several studies (*23–27*). Long short-term memory (LSTM) units are a widely applied (*18*) element of RNNs that help in training a network to learn correlations across many timesteps (*28*).

Additionally, road networks are made up of interconnected road segments that often have strong spatial correlations in traffic volumes and speed. Where data from different road segments is available, convolutional neural networks (CNNs) are promising, as the state-of-the-art approach for many spatial data processing tasks such as image classification (*29*). CNNs have been successfully applied to exploit the spatial road network information in (*24*, *27*, *30*) In fact, in Wang *et al.* (*24*) and Wu *et al.* (*27*), a combination of a convolutional neural network and recurrent neural network were applied to predict the traffic speed and volume, respectively.

In this work, different deep learning architectures to the vehicle speed prediction problem for an arterial road network will be applied. As in previous work, input data was collected from multiple road segments across time which have both spatial and temporal correlations. Unlike in previous work, the authors compare the performance of different architectures for different prediction windows (from 5–30 min ahead). Although the proposed architectures are known and have each been applied for prediction within transportation science, our comparative analysis provides a foundation for future predictive work in selecting appropriate deep network architectures for short- and long-time predictions. The conclusions about which elements of different architectures are effective for our spatio–temporal input data are given, while noting that systematic architecture optimization (for instance over the number of hidden layers, and nodes contained therein) should be performed before stronger conclusions are formed. In the results, the deep learning architectures demonstrate higher performance than traditional prediction methods for all prediction windows for the Sydney road network.

Deep Learning Methodology

The proposed deep learning methodology to predict travel speeds for multiple road segments is presented in this section. There are three main components: feature generation, the multivariate deep learning model development, and model deployment.

Feature Generation

Figure 1 illustrates the data processing and feature generation step. Firstly, the area of interest is identified and road segments with available travel speeds are included. The available data includes spatial data from multiple road segments, and temporal data for each segment. The travel speed data arrives in time-series format, before being processed and transformed into sequential matrices so that the model can learn the spatio–temporal interdependencies of the road segments. As a consequence, the correlations in travel speeds among different road segments are utilized.

Let $\vec{y}$ be the time series of vehicle speeds for a target segment; the model will predict entries of $\vec{y}$ ahead of time. Here $y^{(t)}$ is the target segment's average travel speed for the time interval $(t-1, t]$, and $t \in \{1, 2, ..., n\}$ where n is the total number of time intervals in the dataset. At time t, the target travel speed to be predicted for the following timestep is $y^{(t+1)}$. A "lookback" window of size Δ is defined where the model will use the travel speed data of the last Δ intervals as input for the predictive model. For instance, if $\Delta = 5$, the input vector will include $[y^{(t-5)}, y^{(t-4)}, ..., y^{(t-1)}, y^{(t)}]$. The proposed model also uses Δ intervals of nearby road segments which are spatially related time series as inputs. For example, when forecasting travel speed for a single road segment within 15 road segments in the investigated network (Figure 1), the model also uses data from all other 14 nearby segments as input. These time series are denoted by $\vec{x}_d$, where $d = 1, 2, ..., D$ are indices that label particular road segments. The illustration of $\vec{x}_d$ is presented in Figure 1 as Multivariate Time-Series Travel Speed. Our input vector $\mathbf{X}(t)$ (at time t) includes each road segment with the lookback window Δ, which can be expressed in vector form as

$$\mathbf{X}(t) = [x_1^{(t-\Delta)}, ..., x_1^{(t)}, x_2^{(t-\Delta)}, ..., x_2^{(t)}, ..., x_D^{(t-\Delta)}, ..., x_D^{(t)}, y^{(t-\Delta)}, ..., y^{(t)}]. \tag{1}$$

Alternately, the input can be expressed explicitly in a sequence, as time-incremented rows of a matrix as

$$\mathbf{X}(t) = \begin{bmatrix} x_1^{(t-\Delta)} & x_2^{(t-\Delta)} & \dots & x_D^{(t-\Delta)} & y^{(t-\Delta)} \\ \dots & \dots & \dots & \dots & \dots \\ x_1^{(t)} & x_2^{(t)} & \dots & x_D^{(t)} & y^{(t)} \end{bmatrix}. \tag{2}$$

All features were scaled to the range [0, 1] before being passed to the deep learning model.

Deep Learning Model Development

This section presents the design and development of different deep learning architectures for multivariate travel speed predictions. Different deep learning methods and supporting layers (e.g., dropout) are included in our experiments.

First, some neural network terminology that is necessary to discuss different approaches will be introduced

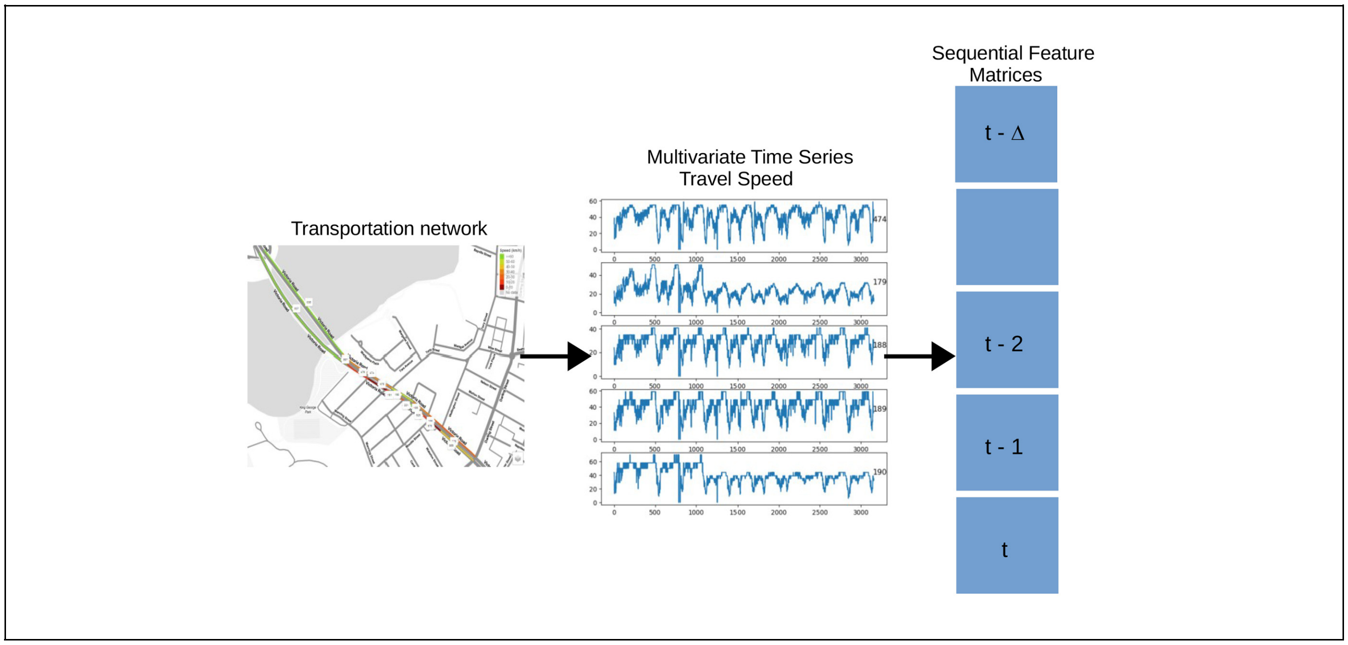
Transportation network
Multivariate Time Series Travel Speed
Sequential Feature Matrices
t - Δ
t - 2
t - 1
t

Figure 1. Feature generation process.

here. More detail and motivation can be found in various reviews, such as (*31–33*). Neural networks are composed of nodes (or artificial neurons) that are connected by edges. Nodes are typically organized in layers, such that the network input is represented by the input layer and the information flows to subsequent layers through the edges. Intermediate layers are known as hidden layers, which pass the information finally to the output layer. The input for a node *j* (beyond the input layer) is given by a weighted sum of outputs from nodes connected to *j*. An *activation function* then acts on each node's input. These basic building blocks are employed to construct different types of processing layers, and several such approaches are described in the following sections.

Convolutional Neural Network. CNNs, which are widely applied to process images, speech, and time series, were first introduced by LeCun *et al*. (*34*). In a modern deep learning architecture, CNN models usually include one or more convolutional layers with nonlinear activation functions and one or more fully connected (or dense) layers. Convolutional layers use spatially restricted connections in a 1-D or 2-D block that have fixed connection weights. These blocks "slide" across the previous layer to compute inputs to the subsequent layer. Each convolutional layer often comes with a sub-sampling (pooling) layer that reduces the size of the data representation, hence reducing the computational requirements as well as minimizing the likelihood of over-fitting (*35*). This pooling operation is performed by sliding an input window across a given layer, and aggregating the values within the window to one single value. This aggregation is often by taking the maximum (Max Pooling). The CNN structure is optionally followed by dense layers, which involve edges from each node in the previous layer to every node in the subsequent layer. These dense layers are similar to standard multilayer neural networks (*36*). Figure 2 illustrates a CNN architecture with two convolutional layers and two sub-sampling (Max Pooling) layers followed by a dense (fully connected) layer. The input layer is presented by sequential feature matrices as described above.

The CNN architecture was originally designed to exploit the 2-D structure of an input image for image processing purposes. This same design can be utilized to treat correlations between travel speeds of multiple related road segments. Since the input includes a spatial dimension and a time dimension (through the lookback window), the convolution could be performed over just the spatial information (1-D convolution) or both dimensions (2-D convolution). A CNN model is capable of learning sequences and capturing the spatial dependency of the dataset (*37*).

Long Short-Term Memory. One major disadvantage of traditional (feed-forward) neural networks in learning time-series patterns is that there is no direct dependency between successive timesteps. LSTM units were proposed (*28*) to learn temporal correlations with long-term dependencies. These LSTM units act as a "memory" to store information. They include nodes with a self-connection along a sequence (i.e., connecting the network across timesteps), along with some additional computation within the unit, and additional unit inputs, and associated weights that must be trained (*28*, *38*). These additional inputs are used as "context" for the effective memory inside the LSTM unit, to control the computation (e.g., the internal state/memory can be written to, preserved, or forget information) as a function of the input. Figure 3 illustrates a standard LSTM, unrolled in time (explicitly represented across different timesteps). The propagation of information across time is represented in Figure 3 by the connection line (horizontal, near the top of the diagram) running through the entire architecture. The output of an LSTM layer depends on the inputs at all previous times.

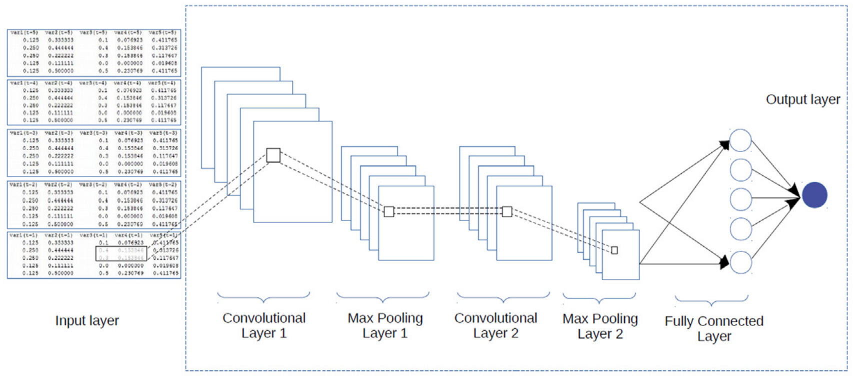

Figure 2. An example of CNN network for multivariate travel speed prediction.

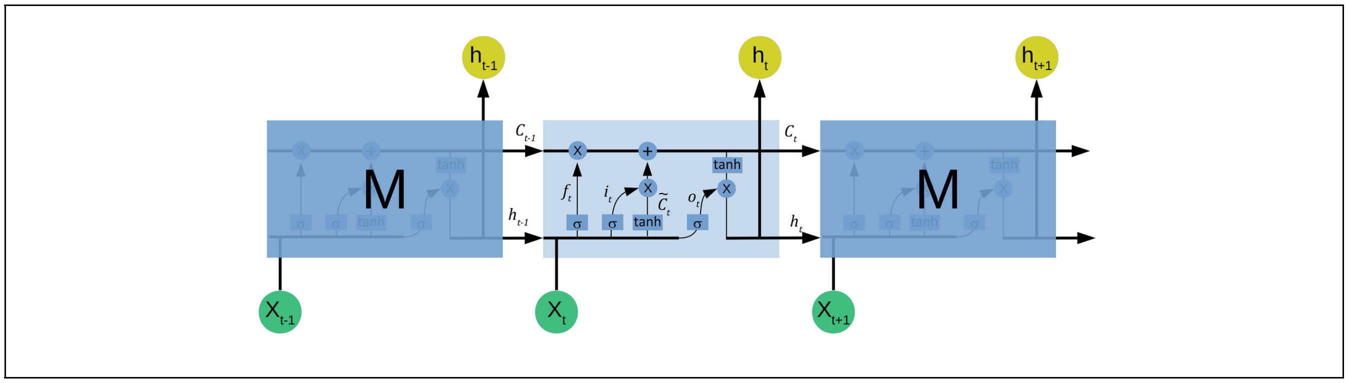

Figure 3. A standard LSTM module with interacting components, unrolled in time.

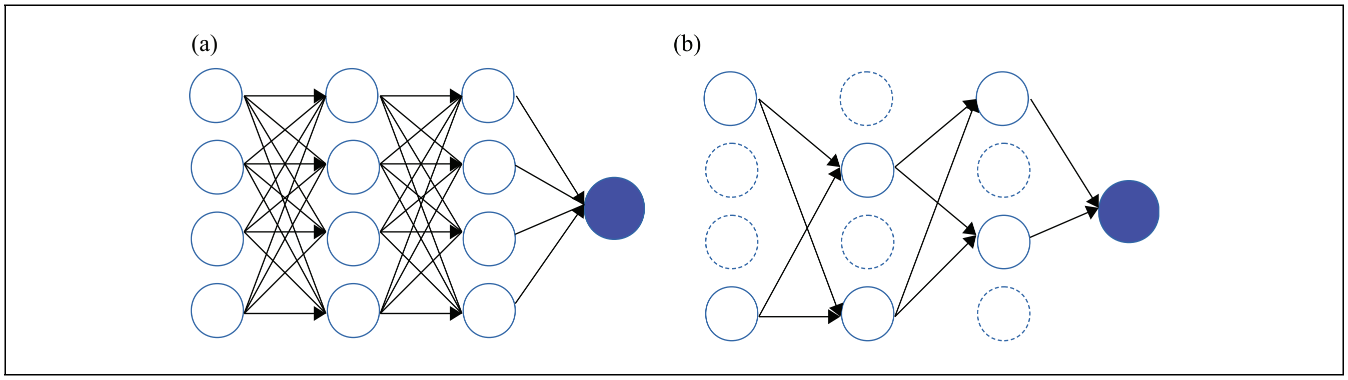

Figure 4. A dense (fully connected) neural network with and without dropout.

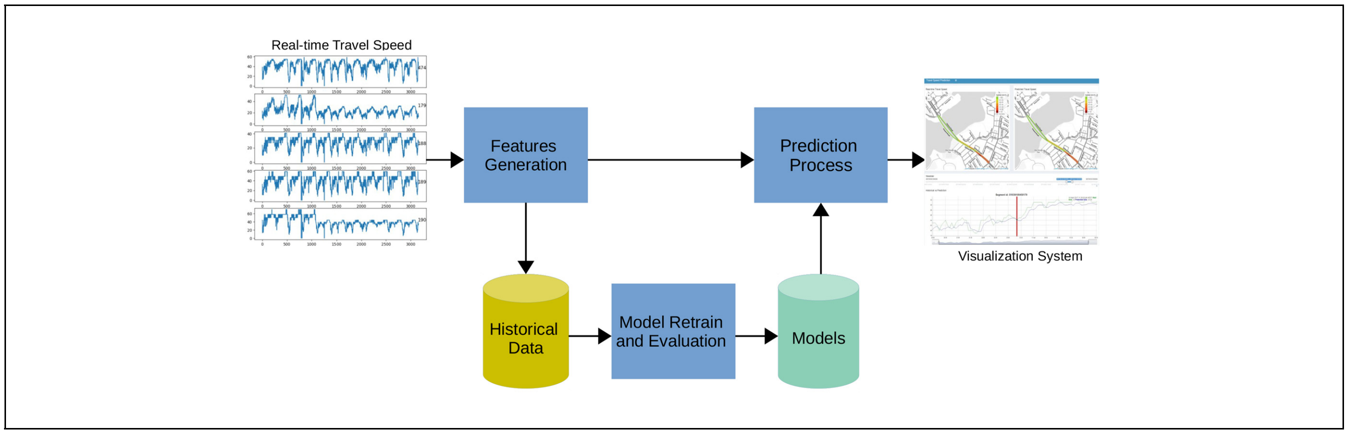

Figure 5. Model deployment and retraining process.

LSTM units were specially designed to capture long-term dependencies, however they lack the analytical power of CNNs for spatial dependencies in the travel speed dataset.

Dropout Layers. In deep learning architectures, the more complexity the network has (e.g., more layers or nodes), the more prone it is to over-fitting. Over-fitting is where the model fits too well to the training data, while performing poorly on unseen data. Dropout can be applied to reduce over-fitting during training (*39*). The key idea is to "drop" some information during training, so that the model is forced to better learn general patterns in the dataset, rather than memorizing specific patterns. This process will compromise some training performance, but improve generalization (performance on unseen data). Figure 4 shows an example of a dense layer with and without dropout. When a dropout layer is applied, during training, a percentage of the outgoing connections from the previous layer are stochastically removed. After training (during testing or application), the model is expected to have learned general patterns, and all connections are henceforth included.

Model Deployment

The models that achieved the best performance are deployed in a decision support system to predict short-term traffic conditions in the observed area. Figure 5 illustrates the real-world application of the iterative model training and deployment process. The real-time travel speeds are periodically retrieved every 5 min and used as input data for the models. To allow the model to perform accurately, the input data needs to be passed through the same features generation module as in the training process. This real-time data is used to generate predictions of upcoming traffic conditions, as well as being collected for model retraining. The predictions are generated by the models in milliseconds, and the outcomes are displayed on the live map to support TMC operators in making precise and timely decisions. A detailed description of the visualization system is provided in the Visualization System section below. The predictive model is retrained and validated frequently with recent traffic data, because the traffic patterns change over time for various reasons (e.g., special events, school holidays, infrastructure updates, etc.). The model could even be updated based on changes in the data, or new model developments. In the proposed system, the models are retrained overnight for daily updates of the traffic patterns. The model update process is referred as "iterative model development" which can be described as follows:

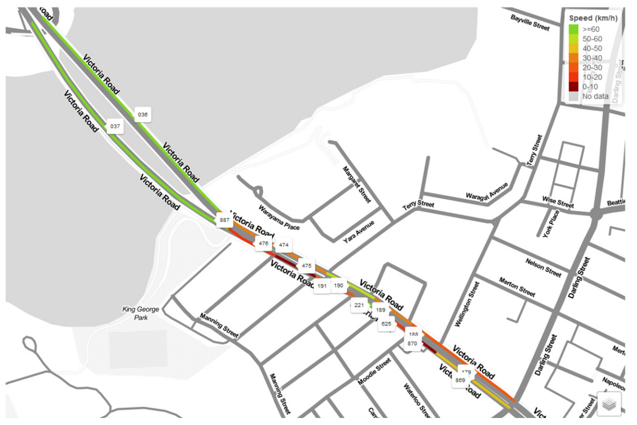

Figure 6. Road segments investigated for travel speed predictions.

1. The features from recent travel speed data are stored in the historical database.
2. The historical database is used to retrain existing models and generate new models by optimizing the parameters of proposed architectures.
3. New models are evaluated and the model with the best performance is deployed for real-time prediction and the visualization system.

Experiments

This section provides details about the data, the model development and the parameter selection processes. The experimental results are also presented and discussed, along with a visual evaluation and real-world application.

Data Description

The investigated area belongs to Victoria corridor, one of the key and very busy corridors in NSW, Australia. The relevant links were chosen from the major Victoria roads segments (for both directions) in Rozelle, NSW where travel speed data is available. This study covers approximately twelve days of travel speed data (from 01-Apr-2017 to 12-Apr-2017) which includes 15 segments of Victoria Road, Drummoyne NSW, Australia. The travel speed data for each segment is recorded every 5 min (e.g., 00:00, 00:05, ..., 23:55). Figure 6 depicts the locations of the investigated road segments on the map, where the 3-digit tags are the segment ids. As presented in this figure, all road segments are directly connected or very close to each other. The travel speeds on these segments are thus expected to have spatial and temporal correlations. However, the data quality is different for each road segment; most of the missing data points were replaced by their values from the previous timestep.

Evaluation Methods

The deep learning models are compared with other popular algorithms in the literature for short-term travel speed prediction. Using the same dataset, the following models have been developed and compared:

- Parametric models: ARIMA and linear regression (LR) (*40*, *41*).
- Non-parametric models: Gaussian process (GP) and k-nearest neighbor (3NN) (*42*, *43*).
- Non-parametric deep learning networks: LSTM- and convolution-based networks.

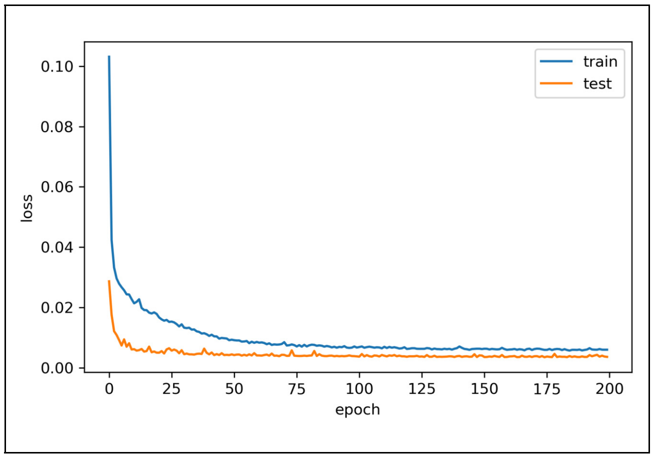

Figure 7. Model performance during training and testing processes for a single run with MSE as loss function.

The travel speed data input for non-deep learning methods is processed and converted into standard time-series format. The general data headers include "Time_index, Segment_Id_1, Segment_Id_2, ..., Segment_Id_15." In this format, the first column contains the time stamp indexes for each 5 min, e.g., "1/04/2017 11:00,""1/04/2017 11:05,""1/04/2017 11:10," etc. The following columns store corresponding travel speeds for individual road segment. An example travel speed record is represented as "1/04/2017 11:00, speed_1, speed_2, ..., speed_15." In some cases, when the data is missing for a specific time stamp, it will be replaced with average travel speed from previous and next time stamp of the same segment. The classical time-series models were developed on Weka (*44*) with the basic heuristic parameters selection process where a set of configurations were defined by the authors and executed by Weka Experiment Environment.

The evaluation metric is the root-mean-square error (RMSE). It is calculated as

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}, \tag{3}$$

where n is the number of predicted values in the sequence (given by the length of the sequence minus the lookback window length). At each time i, $\hat{y}^{(i)}$ is the predicted value and $y^{(i)}$ is the true value.

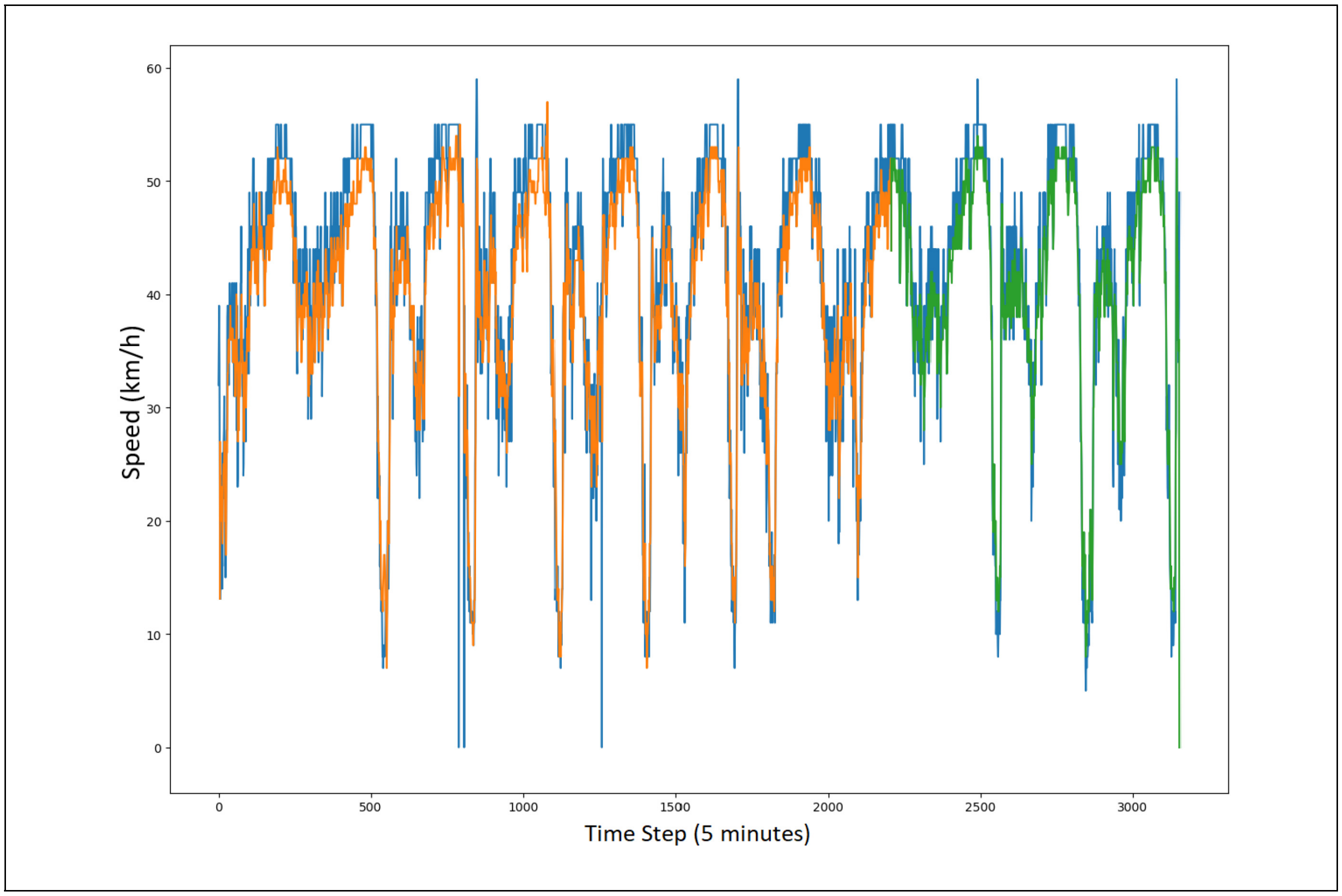

Figure 8. Travel speed predictions for road segment 474 over the training (test) data shown in orange (green). The ground-truth data is shown in blue. These predictions are made 5 min (one timestep) in advance, using the LSTM model.

Model Development and Parameter Selection

The first 8 days of the 12-day dataset is compiled as the training dataset (67% of the data) and the last 4 days is used as the test set (33%). Note that different lengths of training data were separately tested, and the evaluation performance is consistent with more than 7 days of training data. Mean-squared error (MSE) is used as the loss function for model training. This is simply the square of the evaluation metric (RMSE) in Equation 3 above.

Figure 7 represents the performance evaluation during the training and testing procedures. Both the training and testing procedure converge after approximately 100 iterations (epochs). MSE is used as loss function in the training process. The performance difference between training and testing is insignificant, suggesting that the model is not over-fitted to the training data.

Figure 8 shows a snapshot of the LSTM model (described in the following section) fitting results for the training and testing data for road segment number 474. The real data is shown in blue, whereas the (single-run) predictions for the following timestep (5 min ahead) are in orange and green for training and test data, respectively. Despite the high volatility in the data, the LSTM model captures the patterns in the data very well in both the training and testing data. The precise extreme values consistently display errors of around 2km/h; this could be because of the challenge of predicting these precise values. Further testing on the model could evaluate whether additional nodes help with this extreme-value prediction.

The deep learning architectures are implemented using Keras where each method is tested with a number of optimizers (e.g., SGD, Adagrad, Adadelta, Adam, Adamax, Nadam, RMSprop) (*45*). Besides the optimizer (and for a given deep learning architecture), there are two main parameters to be selected during the model training process: the number of epochs and the batch size. In machine learning, an epoch is completed when an entire training set has been passed though the model once. The batch size is the number of elements of the training set that are passed through the model before updating the weights (to gradually optimize the model) based on the model performance over the given batch. For each optimizer, different combinations of epoch and batch size are investigated and a corresponding heat map is generated. Figure 9 presents a heat map visualization of parameter selection for the LSTM model using the Adam optimizer. In this heat map, the test RMSEs are presented in color. Lighter colors denote lower RMSE, corresponding to better performance. The map includes some stochasticity (noise) as just a single run of the model was conducted for each set of parameters, as significant computational resources are required to generate the heat map. When deciding the value of a parameter, both the RMSE value in the parameter evaluation, and the robustness of the parameter (the consistency of surrounding colors) are considered. This “robustness” consideration is motivated by expecting consistent performance of a set of model parameters (with new training and test data inputs) when the nearby model parameter values achieve consistent RMSE results. For instance, 150 epochs and a batch size of 70 performed well for the parameter testing (has low RMSE), however this also has low robustness. For different input data, the model may be expected to perform poorly. In contrast, 250 epochs and a batch size of 65 has both high performance and robustness. The hyperparameters for the model were also selected. The activation functions in the LSTM layer were those predefined by Keras (hyperbolic tangent and hard sigmoid for the recurrent activation), whereas the activation function in the dense output layer was linear. The convolution layers involved 32 output filters (from different convolution “blocks”), a kernel (convolution window) size of 3 and a rectified linear unit (ReLU) activation function. Where dropout was applied, the dropout rate was 0.5.

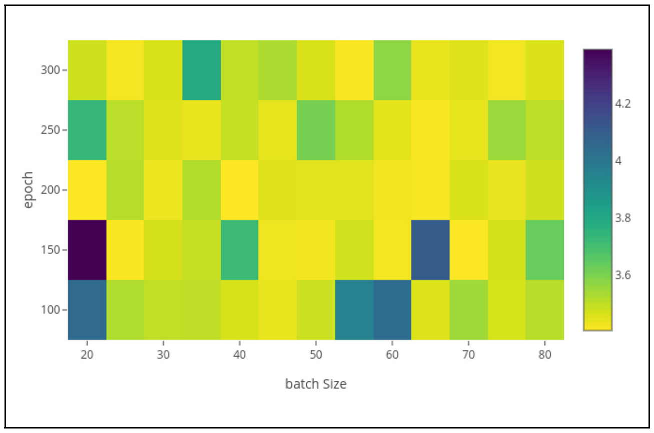

Figure 9. A heat map visualization of parameter selection for the LSTM model. The color scale represents the test RMSE (km/h).

Results and Discussion

To compare the performance of deep learning architectures, as well as alternate predictive models, road segment 474 is selected for evaluations. This segment is in the middle of the investigated area, with a high quality of recorded data (no missing or constant data series). In Table 1, the performance comparison for each model on road segment 474 is presented. The best result for each column is highlighted in bold.

Among classical regression algorithms, LR and GP achieve the best performances with very similar results. For predictions of up to 20 min ahead, the deep learning algorithms consistently outperform classical approaches.

Table 1. Travel Speed Prediction Performances (RMSE)

Model	5 min	10 min	15 min	20 min	25 min	30 min
ARIMA	3.64	4.741	5.476	6.042	6.588	7.133
LR	3.68	4.65	5.27	5.87	6.43	6.97
GP	3.67	4.62	5.25	5.83	6.36	6.88
NN	5.25	5.93	6.39	7.05	7.79	8.7
MLP	4.01	5.8	7.44	9.21	10.96	12.57
LSTM	**3.55** (0.02)	4.48 (0.03)	**5.08** (0.04)	**5.71** (0.04)	**6.33** (0.04)	6.89 (0.02)
CNN-LSTM	3.59 (0.06)	**4.46** (0.06)	5.09 (0.02)	5.80 (0.05)	6.44 (0.07)	7.71 (0.14)
LSTM2						**6.40** (0.12)

Note: Units in km/h; values in brackets are the standard error in the mean.

Table 2. RMSE Scores of the LSTM Model for Travel Speed Prediction

Segment Id	5 mins	10 mins	15 mins	20 mins	25 mins	30 mins
36	3.74	4.94	6.45	7.64	9.22	9.87
37	3.35	3.80	4.17	4.27	4.90	4.59
179	2.00	2.21	2.45	2.91	2.84	2.89
188	2.90	3.08	3.51	3.68	3.94	4.29
189	4.37	5.12	5.63	6.25	6.76	6.94
190	2.43	2.84	3.12	3.31	3.71	4.02
191	4.63	4.95	5.40	5.01	4.92	5.16
221	4.11	4.32	4.19	4.29	4.34	4.60
474	3.50	4.58	5.09	5.54	6.22	6.92
475	0.63	0.59	0.49	0.59	1.90	0.83
476	2.80	2.81	3.82	3.66	3.07	3.58
625	0.60	0.64	0.64	0.64	0.63	0.64
869	1.35	1.30	1.12	1.24	1.14	1.14
870	0.31	0.99	0.27	0.37	0.40	0.32
887	2.56	2.85	3.25	3.79	4.08	3.82
Mean RMSE	2.62	3.00	3.31	3.54	3.87	3.97

Note: Units in km/h.

However, for 25–30 min predictions, the GP method performs equally as well as the LSTM deep learning method, while the CNN-LSTM method has a higher RMSE (results were shown for the test data). The strong performance of the GP model is notable. However, for large numbers of input features as well as large input datasets, computation with the GP approach becomes prohibitively costly (*46*).

The results from three deep learning architectures for travel speed prediction is presented (the lower three entries in Table 1). The model denoted "LSTM" uses a single hidden layer of LSTM units, followed by a dropout layer and a fully connected output node. The input for the model is prepared as in Equation 1, with a look-back window $\Delta = 5$ timesteps (each timestep is 5 min) for 15 relevant road segments. This model is among the top performers for each prediction. Taking the input to this architecture as in Equation 2 did not perform as well. Furthermore, the combination of CNN-LSTM architecture, with a convolutional layer, followed by an LSTM layer (with a dropout layer) before the fully connected output node are also applied. This model uses the input form of Equation 1 (the form of Equation 2 provided similar results), and is intended to exploit the spatial as well as temporal information in the input. Despite the additional complexity, this model was generally outperformed by the simpler LSTM model. This could reflect that the more complex model is more easily trapped in a sub-optimal minimum of the loss function. Alternately, it could reflect that a more complex implementation of spatial convolutions is required. For the CNN layer, the input features are arranged in a regular way (for 1-D convolutions, they are arranged in a list of time-series inputs), and correlations are determined locally. Although different input feature orderings were tested, there is no pre-processing step to explicitly encode which segments were near other segments: such pre-processing could potentially improve the efficacy of the convolution approach, as could additional convolutional layers. Finally, note that the inputs to the LSTM model are densely connected

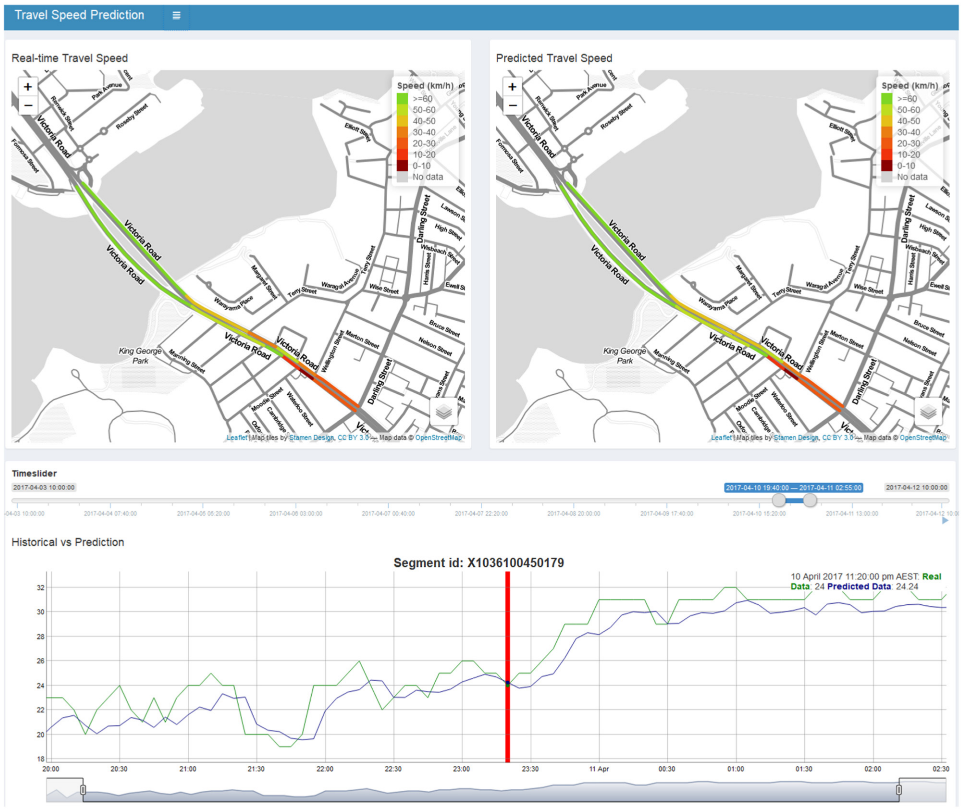

Figure 10. Visualization system for travel speed prediction

to the LSTM input nodes. This means that spatial relationships between features can be learned in this model, even if they are not learned based on the spatial arrangement of the segments in the actual road network—which is expected would help with the generality of the model when testing.

The challenge of making longer-term (30 min) predictions is reflected by the increasing RMSE values for each method. Benefits of the LSTM and CNN-LSTM for the short-term predictions are lost as the prediction time increases; the GP method performs equally well but is also increasingly inaccurate. To address this more challenging prediction timescale, the sequential input structure of Equation 2 is employed for a deeper LSTM network with two hidden layers, "LSTM2." Unlike the other methods, where a lookback window of five timesteps was sufficient for optimal or near-optimal results with fast data processing, the consideration was given to the longest lookback windows for LSTM2. In particular, the result presented is for a lookback window of 24 hours (each timestep is 5 min, so this corresponds to $\Delta = 12 \times 24$ timesteps). The gain from the additional temporal data and temporal processing ability of the deeper LSTM network is evident in the improved result. Note that the calculation time for the corresponding larger dataset is also vastly increased—this is why just the single prediction is included in the results table. For reference, a lookback window of 100 min (20 timesteps) already provided some gain in performance in the same architecture: the mean RMSE in this case was 6.62 (with 0.08 standard error in the mean) for the 30-min prediction.

Table 2 shows the travel speeds predicted by a single run of the LSTM model, from 5 min ahead to 30 min ahead on multiple road segments. The RMSE is presented in each case. Given that the speed limit for each segment is 70km/h, the deep learning model accuracy (RMSE) is lower than 5 km/h. Aside from segment 36, the predictions for 30 min ahead have an RMSE below 7km/h. Note that some segments (e.g., 475 and 870) have extremely low RMSE values for all predictions: this is because of a data quality issue, as these segments have many constant speed values throughout the investigated period, making the time series easier to forecast. On the other hand, segment 36 has the larger RMSE value because of a high level of fluctuations in the time-series data. Typically, only a single run would be performed for prediction of vehicle speeds in real time.

Visualization System and Real-World Application

Visualization System

To support TMC operators to evaluate the model independently, as well as to respond to real-time predictions, the model's prediction results are plotted in an interactive map interface which displays historical data and predicted data simultaneously for visual comparison. Figure 10 illustrates the visualization system for third-party evaluation. The system is developed using Shiny on R platform. In this interface, the top left map displays real traffic conditions which are color-coded based on historical travel speed of each road segment. The map on the right side shows corresponding prediction results on the same area. The operator can choose to validate the predictions from 5–30 min ahead for any given data point in the period.

The historical data and prediction distribution for any sub-period (e.g., 1 hour, 1 day, 1 week, etc.) can be reviewed easily to understand the traffic patterns before and after the prediction time. This function is also useful to look back and analyze any abnormal traffic patterns for road accidents or planned events. Furthermore, using the play-back button on the right-hand corner of the timeline, the traffic conditions can be updated dynamically, including the color-coded segments on the map and traffic speed plots which are displayed and automatically updated over time.

Real-World Application

As demonstrated in the Results section, the proposed system is capable of forecasting the travel speeds on multiple segments simultaneously for up to 30 min ahead.

One key application of these results is that they can be used to manage incident response. The model can predict and help analyze the impact of an incident on nearby road segments, as soon as an incident is reported. The travel speed prediction and visualization system is expected to help the TMC operators to improve their procedures, response plans, and resource allocations to traffic incidents, to quickly reduce congestion and improve road safety.

From the experiments, as the prediction took only a few milliseconds on an Intel i7-7700 machine with 8 cores and 16GB of RAM, the system is fast enough to run using continuous, live travel speeds (updating every 5 min) for a suburb's main roads, to consistently predict the traffic conditions up to 30 min ahead. As a consequence, it can be utilized to monitor a real-time traffic network for a specific area (e.g., areas with special events or a traffic accident) and support timely planning for unusual traffic patterns that may arise.

Conclusions

This paper introduces a decision support system for short-term travel speed prediction on multiple road segments. A number of deep learning models, including RNNs with LSTM units and convolutional components, have been investigated to capture the temporal and spatial correlations between the travel speeds on spatially related road segments. Experimental results demonstrate that the deep neural networks outperform other traditional approaches in short-term travel speed prediction on multiple road segments. The best-performing models (LSTM for up to 25 min and LSTM2 for 30 min) are then integrated into a visualization system for real-world application.

The system can be directly applied for real-time decision support at TMCs, and can provide a real-time forecast of travel speed on multiple road segments up to 30 min in advance.

Acknowledgments

This study was supported by Strategic Research Funding at the Data61, CSIRO, Australia, and partly by the NSW Premier's Innovation Initiative project sponsored by Transport NSW.

Author Contributions

The authors confirm contribution to the paper as follows: study conception and design: HN, CB, LMK, CC; data collection: HN; analysis and interpretation of results: HN, CB, YF; draft manuscript preparation: HN, CB, LMK. All authors reviewed the results and approved the final version of the manuscript.

References

1. Wu, C.-H., C.-C. Wei, D.-C. Su, M.-H. Chang, and J.-M. Ho. Travel Time Prediction with Support Vector Regression. *Proc., Intelligent Transportation Systems, 2003*. IEEE, Vol. 2, 2003, pp. 1438–1442.
2. De Fabritiis, C., R. Ragona, and G. Valenti. Traffic Estimation and Prediction Based on Real Time floating Car Data. *11th International IEEE Conference on Intelligent Transportation Systems, 2008*, IEEE, 2008, pp. 197–203.
3. Innamaa, S. Short-term Prediction of Travel Time Using Neural Networks on an Interurban Highway. *Transportation*, Vol. 32, No. 6, 2005, pp. 649–669.
4. D'Angelo, M., H. Al-Deek, and M. Wang. Travel-time Prediction for Freeway Corridors. *Transportation Research Record: Journal of the Transportation Research Board*, 1999. 1676: 184–191.
5. Yao, B., C. Chen, Q. Cao, L. Jin, M. Zhang, H. Zhu, and B. Yu. Short-term Traffic Speed Prediction for an Urban Corridor. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 32, No. 2, 2017, pp. 154–169.
6. Oh, S., Y.-J. Byon, K. Jang, and H. Yeo. Short-term Travel-time Prediction on Highway: A Review of the

Data-driven Approach. *Transport Reviews*, Vol. 35, No. 1, 2015, pp. 4–32.

7. Vlahogianni, E. I., M. G. Karlaftis, and J. C. Golias. Short-term Traffic Forecasting: Where We are and Where We're Going. *Transportation Research Part C*, Vol. 43, 2014, pp. 3–19.
8. Billings, D., and J.-S. Yang. Application of the ARIMA Models to Urban Roadway Travel Time Prediction-A Case Study. *SMC'06. IEEE International Conference on Systems, Man and Cybernetics, 2006*, IEEE, 2006, Vol. 3, pp. 2529–2534.
9. Khoei, A. M., A. Bhaskar, and E. Chung. Travel Time Prediction on Signalised Urban Arterials by Applying SARIMA Modelling on Bluetooth Data. In *36th Australasian Transport Research Forum (ATRF)* 2013, 2013.
10. Sun, H., H. X. Liu, H. Xiao, R. R. He, and B. Ran. Use of Local Linear Regression Model for Short-Term Traffic Forecasting. Transportation Research Record: Journal of the Transportation Research Board, 2003. 1836: 143–150.
11. Haworth, J., J. Shawe-Taylor, T. Cheng, and J. Wang. Local Online Kernel Ridge Regression for Forecasting of Urban Travel Times. *Transportation Research Part C: Emerging Technologies*, Vol. 46, 2014, pp. 151–178.
12. Idé, T., and S. Kato. Travel-time Prediction Using Gaussian Process Regression: A Trajectory-based Approach. *Proc., 2009 SIAM International Conference on Data Mining*, SIAM, 2009, pp. 1185–1196.
13. Myung, J., D.-K. Kim, S.-Y. Kho, and C.-H. Park. Travel Time Prediction Using k Nearest Neighbor Method with Combined Data from Vehicle Detector System and Automatic Toll Collection System. *Transportation Research Record: Journal of the Transportation Research Board*, 2011. 2256: 51–59.
14. Dia, H. An Object-oriented Neural Network Approach to Short-term Traffic Forecasting. *European Journal of Operational Research*, Vol. 131, No. 2, 2001, pp. 253–261.
15. Park, D., L. R. Rilett, and G. Han. Spectral Basis Neural Networks for Real-time Travel Time Forecasting. *Journal of Transportation Engineering*, Vol. 125, No. 6, 1999, pp. 515–523.
16. Van Lint, J., S. Hoogendoorn, and H. Van Zuylen. Freeway Travel Time Prediction with State-space Neural Networks: Modeling State-space Dynamics with Recurrent Neural Networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2002. 1811: 30–39.
17. Dharia, A., and H. Adeli. Neural Network Model for Rapid Forecasting of Freeway Link Travel Time. *Engineering Applications of Artificial Intelligence*, Vol. 16, No. 7-8, 2003, pp. 607–613.
18. Ali, U., and T. Mahmood. Using Deep Learning to Predict Short Term Traffic Flow: A Systematic Literature Review. *Intelligent Transport Systems – From Research and Development to the Market Uptake* (T., Kováčiková, b. Buzna, G. Pourhashem, G. Lugano, Y. Cornet, and N. Lugano, eds.), Springer International Publishing, Cham, 2018, pp. 90–101.
19. Lv, Y., Y. Duan, W. Kang, Z. Li, and F. Y. Wang. Traffic Flow Prediction with Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 2015, Vol. 16, No. 2, pp. 865–873.
20. Koesdwiady, A., R. Soua, and F. Karray. Improving Traffic Flow Prediction with Weather Information in Connected Cars: A Deep Learning Approach. *IEEE Transactions on Vehicular Technology*, Vol. 65, No. 12, 2016, pp. 9508–9517.
21. Jia, Y., J. Wu, and Y. Du. Traffic Speed Prediction Using Deep Learning Method. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1217–1222.
22. Polson, N. G., and V. O. Sokolov. Deep Learning for Short-term Traffic flow Prediction. *Transportation Research Part C: Emerging Technologies*, Vol. 79, 2017, pp. 1–17.
23. Ma, X., H. Yu, Y. Wang, and Y. Wang. Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory. *PLoS One*, Vol. 10, No. 3, 2015, pp. 1–17.
24. Wang, J., Q. Gu, J. Wu, G. Liu, and Z. Xiong, Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 499–508.
25. Yu, R., Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, *Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting*, 2017, pp. 777–785.
26. Zhao, Z., W. Chen, X. Wu, P. C. Y. Chen, and J. Liu. LSTM Network: A Deep Learning Approach for Short-term Traffic Forecast. *IET Intelligent Transport Systems*, 2017.
27. Wu, Y., H. Tan, L. Qin, B. Ran, and Z. Jiang. A Hybrid Deep Learning Based Traffic flow Prediction Method and its Understanding. *Transportation Research Part C: Emerging Technologies*, Vol. 90, 2018, pp. 166–180.
28. Hochreiter, S., and J. Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780.
29. Krizhevsky, A., I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
30. Ma, X., Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors*, Vol. 17, No. 4, 2017.
31. Lipton, Z. C., J. Berkowitz, and C. Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019v4*, 2015.
32. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks*, Vol. 61, 2015, pp. 85–117.
33. LeCun, Y., Y. Bengio, and G. Hinton. Deep Learning. *Nature*, Vol. 521, 2015, pp. 436–444.
34. LeCun, Y., and Y. Bengio. Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks*, Vol. 3361, No. 10, 1995, p. 1995.
35. Scherer, D., A. Müller, and S. Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. *International Conference on Artificial Neural Networks*, Springer, 2010, pp. 92–101.

36. Hornik, K., M. Stinchcombe, and H. White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366.
37. Ke, J., H. Zheng, H. Yang, and X. M. Chen, Short-term Forecasting of Passenger Demand Under On-demand Ride Services: A Spatio-temporal Deep Learning Approach. *Transportation Research Part C: Emerging Technologies*, Vol. 85, 2017, pp. 591–608.
38. Kalchbrenner, N., I. Danihelka, and A. Graves. Grid Long Short-term Memory. *arXiv preprint arXiv:1507.01526*, 2015.
39. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, Vol. 15, No. 1, 2014, pp. 1929–1958.
40. Milenković, M., Z. Avramović, M. , L. Švadlenka, V. Melichar, and N. Bojović. SARIMA Modelling Approach for Railway Passenger flow Forecasting. *Transport*, 2016, pp. 1–8.
41. Neter, J., W. Wasserman, and M. H. Kutner. *Applied Linear Regression Models*. 2nd ed., Richard D. Irwin, Inc., 1989.
42. Rasmussen, C. E. Gaussian Processes in Machine Learning. *Advanced Lectures on Machine Learning*, 2004, pp. 63–71.
43. Altman, N. S. An Introduction to Kernel and Nearest-neighbor Nonparametric Regression. *The American Statistician*, Vol. 46, No. 3, 1992, pp. 175–185.
44. Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, Vol. 11, No. 1, 2009, pp. 10–18.
45. Chollet, F. *Keras: The Python Deep Learning Library*, 2015. https://keras.io/.
46. Rasmussen, C. E., and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

The Standing Committee on Artificial Intelligence and Advanced Computing Applications (ABJ70) peer-reviewed this paper (19-03040).