Real-time Signal Queue Length Prediction Using Long Short-Term Memory Neural Network

Rezaur Rahman and Samiul Hasan, Ph.D.
University of Central Florida
Department of Civil, Environmental, and Construction Engineering
Orlando, Florida, USA
E-mail: samiul.hasan@ucf.edu

ABSTRACT
Optimal traffic control and signal planning can significantly reduce traffic congestion and potential delays at intersections. However, a major challenge to optimize traffic signal timing is to accurately predict traffic before commencing the next cycle. An optimal strategy cannot be achieved with a poor prediction of future traffic. In this study, using a deep learning approach, we develop a data-driven real-time queue length prediction technique. We consider a connected corridor where information from vehicle detectors (located at the intersection) will be shared to consecutive intersections. We assume that the queue length of an intersection in the next cycle will depend on the queue length of the target and two upstream intersections in the current cycle. We use InSync adaptive traffic control system (ATCS) data to train a Long Short-Term Memory Neural Network model capturing time-dependent patterns of a queue of a signal. To avoid overfitting and select the best combination of hyperparameters, we use sequential model-based optimization (SMBO) technique. Our experiment results show that the proposed model performs very well to predict the queue length. Although we run our experiments predicting the queue length for a single movement, the proposed method can be applied for other movements as well.

Index Terms— Adaptive Traffic Control System, Control parameters, Data-driven, Real-time queue length, Long Short-Term Memory Neural Network.

1. Introduction

Traffic congestion is a serious problem in most of the urban areas. In 2011, it caused urban residents in the USA to spend 6.9 billion hours more in traveling and cost them an extra 3.1 billion gallons of fuel, for a congestion cost of $160 billion (*1*). Inefficiencies in traffic signal timing due to poor green time allocation, inability to respond quickly to real-time conditions, and lack of coordination between adjacent intersections are a few major causes of congestion problem (*2*). To effectively manage traffic in a signal, researchers are testing innovative traffic control systems based on real-time traffic flows.

For instance, Adaptive Traffic Control System (ATCS) is a state-of-the-art system and a major component of intelligent transportation system (ITS) to efficiently manage and distribute traffic in real time. In general, an ATCS consists of three major components: vehicle detection and information gathering, future traffic prediction (traffic flow, queue length etc.), and optimization of the signal timing and phasing. Adaptive traffic signal controllers mostly rely on infrastructure-based sensors or video cameras to detect the vehicles and estimate vehicular queue length and delay. For example, InSync adaptive traffic signal control systems receive inputs related to queue volume, occupancy, and delay in real time via video-based sensors, sometimes assisted with

[Accepted version of the article published in Neural Computing and Applications. The published article is available at https://link.springer.com/article/10.1007/s00521-020-05196-9]

existing roadway detectors. Receiving the inputs of actual traffic demand, the system determines phase priority based on estimated occupancy and delay (*3*). This complex decision-making process happens continuously in real time.

However, detector based systems have several limitations: they only provide instantaneous position of a vehicle rather than direct measurement of traffic (speed, location) states; they have predefined detection zones which only estimate queues that are shorter than the distance between vehicle detector and intersection stop line (*4*); and the detection system consists of multiple sensors, hence the installation and maintenance cost is considerably high (*5*). Also, the overall system has to process all the information in real time, causing erroneous vehicle detection and queue estimation.

Emerging vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication systems have the potential to improve existing ATCS technologies, reducing their dependency on video-based sensors and roadway detectors. Communication between vehicle and infrastructure enables the traffic controller to acquire information regarding vehicle's position, queue length, and delay in real time with better accuracy (*6*). However, availability of such multiresolution data creates a new challenge: how to process these high dimensional spatio-temporal information to predict future traffic (e.g., queue length) for different time horizons (*7*). Optimal traffic operations largely depend on the accuracy of the predicted queue length. Before commencing the next cycle, a traffic controller depends on the predicted queue length to optimize the cycle length. To predict the queue length for future cycle, we need a robust method that can deal with sharp non-linearities in traffic flow.

In recent years, data-driven methods have widely been applied for solving numerous problems such as text processing and classification (*8*, *9*), image and video processing (*10*, *11*), cloud computing (*12*) etc. Similarly, data-driven approaches are receiving more attention in traffic analysis due to the availability of multi-resolution traffic data. Transportation researchers are exploring data-driven approaches to solve different transportation problems, since these approaches are easy to deploy in a real-time context.

A few commonly used data-driven approaches include support vector machine (SVM) (*13*, *14*), *k*-nearest neighbor (KNN) (*15*), artificial neural network (ANN) (*16*, *17*), ARIMA (*18*) etc. These models perform reasonably well for predicting traffic states (speed, travel time, traffic flow etc.) (*19–26*). A deep learning model is one of the most recent innovations in machine learning. It captures sharp discontinuities in traffic flows using non-linear functions (tanh, sigmoid etc.) (*27*). Applications of deep learning and neural computing in transportation have allowed us to deal with more complex problems and big data (*28–32*).

In this study, we consider a corridor of intersections where consecutive intersections will share information with each other (infrastructure to infrastructure communication) and gather information of upcoming vehicles (vehicle to infrastructure communication). We develop a data-driven approach to predict lane-based queue length for an intersection. We anticipate that with emerging connected vehicles technologies and road environments, information (traffic state, queue length etc.) from one intersection will be easily available to another intersection. For our experiments, we use InSync adaptive signal data which provides queue lengths and wait times (time required for the first vehicle to clear the intersection) for different vehicular movements. We implement a Long-Short Term Memory Neural Network (LSTM-NN) model to predict the queue length for the next cycle based on queue length and wait time of three consecutive intersections at

current cycle. To avoid any overfitting, we apply sequential model-based optimization (SMBO) for selecting appropriate dropout at different stacked layers. Finally, we run the experiments to predict queue lengths for north through traffic. Same method can be applied to predict queue lengths for all the other movements as well.

This method is an initial step toward data-driven traffic control and optimization using real-time traffic data. This study has made several contributions:

- it develops a data-driven method for signal queue prediction utilizing real-world traffic data; traditional approaches rely on simulations that fail to deal with real-world traffic queue variations.
- it demonstrates the potential of implementing coordination among traffic signals in a connected vehicle environment; this can help us predict queue lengths and coordinate among signals for a large number of intersections without deploying costly adaptive traffic control technologies.
- it provides empirical evidences that SMBO algorithm can help us design a task-specific neural network architecture by estimating optimal hyperparameters.

2. LITERATURE REVIEW

Vehicular queue length estimation is a crucial step in optimizing signal plans (*33–35*) and measuring the performance of a signalized intersection (*36*). Especially for ATCS technologies, the signal control logic is based on queue lengths estimated in real time. Researchers have developed several methods to estimate queue lengths for traffic signals using loop detector data and signal timing information. These studies can be divided into two categories. The first one is based on the analysis of cumulative input output to a signal link, proposed by Webster (*37*) and later improved by several researchers (*33*, *38–41*). In this method, queue length is derived from cumulative arrivals and departures of an intersection. However, this method is effective in determining the queue length formation process or effective queue size, but not sufficient to obtain the spatial distribution of queue length for a given time (*42*). Moreover, application of this approach is limited, since cumulative input-output methods can only be applied when the queue length does not exceed the vehicle detector location (*4*). The second category is based on shockwave analysis: how a queue forms and dissipates at an intersection. Lighthill and Whitham (*43*) first demonstrated shockwave theory for uninterrupted flow for a long stretch of arterial road. Later Richards (*44*) proposed a similar theory for traffic flow analysis considering the effect of traffic signal on traffic stream behavior. Stephanopolos and Michalopoulos (*42*) expanded it for isolated signalized intersections to develop analytical expressions for the maximum queue length estimation.

Recent innovation in vehicle detection and sensing technologies enables us to collect multi-resolution traffic data in real time. Consequently, real-time queue length estimation such as cycle by cycle queue length, has gained more attention. Several studies have proposed shockwave theory based approaches for real-time queue estimation utilizing traffic signal timing and loop detector data (*4*, *36*, *45*, *46*). Moreover, mobile traffic sensors, such as GPS equipped probe vehicles, cellular phones, connected vehicles, and other tracking devices, provide an alternative to fixed-location sensors for real-time queue estimation (*46–50*). Especially, emerging connected vehicle technologies provide precise information on vehicle position and traffic state variations in real time, potentially improving the queue length estimation technique (*50–52*).

Although ubiquitous sensing devices and emerging connected vehicle technologies provide

multiresolution traffic data, data-driven techniques for estimating queue length are less common. Chang and Su (*53*) developed a data-driven neural network model for predicting queue length at short time step (3s). However, to train their models, they used data from simulation experiments. Lee et al. (*54*) explored a neural network based deep learning model for queue length estimation. They also used traffic simulations to generate the training data. As such, these studies have two limitations: *first*, these approaches do not consider the coordination among multiple intersections; *second*, they have not been validated with real-world traffic data.

Recently, Gan et al. (*55*) applied Artificial Neural Network based method to predict the trajectory length of manually controlled ships for optimal signal control in river traffic management. Here, we develop a new approach to predict the queue lengths at an intersection to optimize traffic signal parameters in the context of urban traffic management. Here, we consider a connected corridor (e.g., via V2I communication) where information is shared between different traffic control devices, giving a better picture for the overall traffic state variations. In such cases, a robust method is needed to predict signal queue lengths considering both short-term and long-term dependencies in the traffic flow patterns.

To solve this problem, we trained a Long-Short Term Memory Neural Network (LSTM-NN) model to predict the queue length for the next cycle based on queue length and wait time of three consecutive intersections in the current cycle. Two challenges of a neural network based approach include optimal hyperparameter selection (*56*, *57*) and overfitting. To overcome these challenges, we applied sequential model-based optimization (SMBO) for selecting appropriate dropout at different stacked layers.

In general data driven approaches reduces the complexity of queue estimation compared to traditional traffic flow theory based approaches. To model queue formulation, traffic flow theory based approaches rely on certain assumptions on vehicle distributions, vehicle arrival rates, and lane changing probability (*58*). Whereas data driven approaches rely on historical traffic patterns and queue length variations to address these factors. Hence, these methods are more reliable for real world applications. The proposed method and the case study are an initial step towards data-driven traffic control and optimization using real time traffic data.

3. Methodology

3.1. Long-Short Term Memory Neural Network

Deep learning has created a unique opportunity to deal with more complex problems in transportation. The emergence of deep learning methods has been encouraged by a tremendous increase in computational power and data availability. Artificial Neural Network (ANN), the common deep learning technique, is not suitable to deal with complex time series problems due to its inability to capture the inter-dependency among the hidden states while processing sequential data (*59*). To solve this issue, ANN was extended to model data with temporal or sequential structure and varying input lengths, known as Long Short Term Memory Neural Network (LSTM NN) (*60*).

An LSTM NN is composed of three layers: input layer, recurrent hidden layer, and output layer. The basic unit of the hidden layer is a memory block, which makes it different from traditional neural networks. Memory block contains memory cell which is divided into two parts: short-term state or hidden state $(h_{(t)})$ and long-term state$(c_{(t)})$. The short-term state is the output of a cell at a given time step (t) and the long-term state $(c_{(t)})$ stores the information to capture the

long-term dependencies among the current hidden state (cell state) and the previous hidden states (cell states) over time. Traversing from the left to the right, the long-term state passes through a forget gate and drops some memories and then adds some new memories via an addition operation.

As shown in Fig. 1, a fully connected LSTM cell contains four layers (sigma and tanh) and the input vector ($X_{(t)}$) (historical input data such as queue length and wait time at consecutive intersections) and the previous short-term state ($h_{(t-1)}$) are fed into these layers. The main layer uses tanh activation functions which outputs ($g_{(t)}$). The output from this layer is partially stored in the long-term state ($c_{(t)}$). The other three layers are gate controllers using logistic activation functions and their output ranges from 0 to 1. The forget gate $f_{(t)}$ controls which parts of the long-term state should be erased, while input gate $i_{(t)}$ decides which parts of the input should be added to the long-term state. The output gate $o_{(t)}$, finally controls which parts of the long-term state should be read and gives output $y_{(t)}$ (predicted queue length for next time step) at a given time step (t). The mathematical formulation of these operations is adapted from (*61*):

$$\text{Input gate: } i_{(t)} = \sigma\left(W_{xi}^T . X_{(t)} + W_{hi}^T . h_{(t-1)} + b_i\right) \quad (1)$$

$$\text{Forget gate: } f_{(t)} = \sigma\left(W_{xf}^T . X_{(t)} + W_{hf}^T . h_{(t-1)} + b_f\right) \quad (2)$$

$$\text{Output gate: } o_{(t)} = \sigma(W_{xo}^T . X_{(t)} + W_{ho}^T . h_{(t-1)} + b_o) \quad (3)$$

$$\text{Cell input: } g_{(t)} = \tanh(W_{xg}^T . X_{(t)} + W_{hg}^T . h_{(t-1)} + b_g) \quad (4)$$

And, the long-term and short-term states are calculated using the following equations (*61*),

$$\text{Long-term state: } c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \quad (5)$$

$$\text{Short-term state: } y_{(t)} = h_{(t)} = o_{(t)} \otimes tanh(c_{(t)}) \quad (6)$$

where, $W_{xi}, W_{xf}, W_{xo}, W_{xg}$ are the weight matrices of each of the four layers for their connection to the input vector X_t. $W_{hi}, W_{hf}, W_{ho}, W_{hg}$ are the weight matrices of each of the four layers for their connection to the short-term state (h_{t-1}) and b_i, b_f, b_o, b_c are the bias terms for each of the four layers. $o_{(t)}$ and $g_{(t)}$ represent the output from the output layer and cell input layer. $\sigma(.)$ represents the sigmoid function $\frac{1}{1+exp(-x)}$ and $\tan h\,(.)$ represents the hyperbolic tangent function $\frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$.

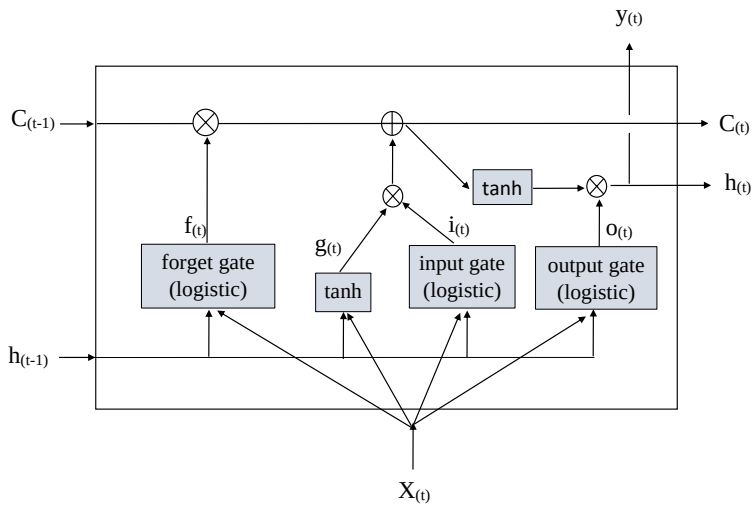


Fig. 1. The complete structure of an LSTM cell

3.2. Hyperparameter Selection

Selecting the best combination of hyperparameters for a given task is a critical step in developing deep neural networks. Especially, generating a task-specific architecture for a neural network based model is more challenging (*62*). Moreover, deep learning models suffer from overfitting which can lead to erroneous results. Several methods such as grid search and random search (*63*) have been widely used for hyperparameter selection for different models. However, the parameter selection and optimization methods largely depend on the underlying modeling framework. For example, in case of Interval Type-2 Fuzzy Neural Networks, big bang-big crunch (BBBC) optimization and particle swarm optimization (PSO) showed better performance in selecting best membership functions (*56*). Whereas, in case of Dendritic Neuron Model, the best combination of parameters was selected using Taguchi's experimental design method (*57*).

In this study, to obtain the best combination of hyperparameters, we apply the sequential model based optimization (SMBO) method (*63–65*) with tree-structured Parzen estimator (TPE) algorithm. The SMBO method sequentially constructs models to approximate the performance of a given set of hyperparameters based on historical measurements, and then subsequently chooses new hyperparameters and tests the performance of the model. The historical measurements include the records of the set of hyperparameters previously used and corresponding value of the objective function. The objective function represents a single valued score that we want to minimize such as mean squared error. Finally, the TPE algorithm finds the optimal values of the hyperparameters. In this algorithm, the hyperparameter space is considered as a tree, where the value chosen for one hyperparameter determines what hyperparameter will be chosen next and what values are available for it.

SMBO optimization method, adopted from (*63)*, approximates the true fitness function $f: \chi \rightarrow \mathrm{R}$ with a surrogate function which is easier to evaluate. Here the surrogate function is based on expected improvement (EI_{y*}) which can be defined as (*66*),

$$EI_{y*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy \tag{7}$$

Where, y^*represents the threshold value of the objective function, x represents the proposed set of

hyperparameters, y represents the actual value of the objective function using a set of hyperparameters x, and $p(y|x)$ is the surrogate probability model expressing the probability of y given x. The aim is to maximize the expected improvement with respect to x using the TPE algorithm.

TPE algorithm numerically optimizes the surrogate function to obtain optimal hyperparameters x^* that maximize the expected improvement. The algorithm builds a model applying the Bayesian rule instead of directly representing $p(y|x)$.The formulation is written as follows,

$$p(y|x) = \frac{p(x|y)*p(y)}{p(x)} \tag{8}$$

where, model $p(x|y)$ represents the probability of the hyperparameters given the objective function value. The TPE defines $p(x|y)$ using two densities:

$$p(x|y) = \begin{cases} l(x), \ when\ y < y^* \\ g(x), \ when\ y \geq y^* \end{cases} \tag{9}$$

The TPE algorithm chooses some quantile q of the observed value y in such a way that $p(y < y^*) = q$, where the threshold value, y^* is greater than the largest value of y in q. So, we have two distributions for the hyperparameters: (1) $l(x)$, if the objective function is less than the threshold and (2) $g(x)$, if the objective function is greater than the threshold. Intuitively, we want the hyperparameters to be in $l(x)$, because that will reduce the objective function value and increase the expected improvement. After few substitutions, the integral in equation (7) converges into,

$$EI_y^{*(x)} \propto \left(q + \frac{g(x)}{l(x)}(1-q)\right)^{-1} \tag{10}$$

So, the expected improvement is directly proportional to $\frac{l(x)}{g(x)}$. This indicates the hyperparameters which are more likely to be under $l(x)$ and less likely to be under $g(x)$ will maximize the expected improvement. The algorithm iteratively chooses a set of candidates x^* that is more likely to be in the first group and less likely to be in the second one. Finally, it outputs the optimal set of hyperparameters that yields the maximum expected improvement.

We implement the SMBO optimization method using hyperopt library (*64*). The hyperopt library gives the ability to define a prior distribution for each parameter. Table 1 prsents the initial values of the parameters, selected based on previous studies and our preliminary observations. From the previous studies (*32*, *67*), we find that if the input features have a lower number of dimensions, a single layer LSTM model shows superior performance. However, we experiment with both single layer and stacked LSTM layers to check which one works best for this particular problem. To select the activation function for different layers, we consider most non-linear functions widely used in previous studies. One of the critical steps for designing an appropriate architecture for neural network is selecting the dropout value to avoid any overfitting of the model. We initialize the dropout value from a uniform distribution ranging from 0 to 1. Usually, batch size does not influence the model performance; rather it influences the total memory requirement and training time. We randomly assign the batch size based on different time sequence length (one day, two day etc.). Finally, in case of optimizer, we consider all the optimization algorithms for neural network based model with default learning rates.

Table 1 Initial distribution of each parameter

Parameter Name	Distribution	Values
Number of Stacked LSTM layers	Categorical	$x \in \{1,2\}$
Activation Function in each layer	Categorical	$x \in \{relu, tanh, sigmoid\}$
Number of Units in the First Layer	Categorical	$x \in \{64,128,256,512\}$
Number of Units in the Second Layer	Categorical	$x \in \{64,128,256,512\}$
Dropout in each layer	Uniform	$x \in [0,1]$
Optimizer	Categorical	$x \in \{adam, sgd, adagrad, rmsprop\}$
Batch Size	Categorical	$x \in \{360,720,1440\}$

3.3. LSTM-NN Framework for Queue Length Prediction

In this study, we assume that for a given intersection, the queue length for a specific movement will depend on that intersection and upstream intersections. For example, north through (NT) for the next cycle (t+1) will depend on the queue length and vehicle wait time of that intersection and the adjacent upstream intersections at current cycle (t). As input vectors, we have added the upstream intersections and target intersection queue length and wait time ($X(t) = [q_{i-2}(t), q_{i-1}(t), q_i(t)\ w_{i-2}(t), w_{i-1}(t), w_i(t)]$) (Fig. 2).

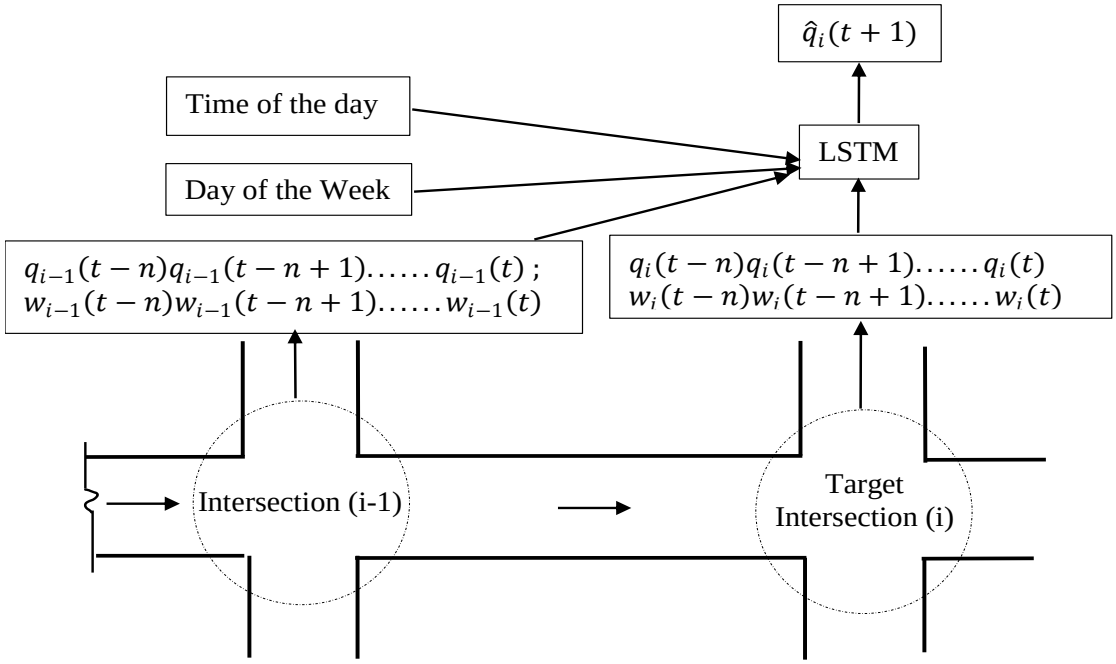


Fig. 2. The layout of the Variables for Prediction

Moreover, to capture this temporal influence, we add the time of the day and day of the week as independent variables. In a regular traffic scenario, we observe that the daily variation of traffic flow follows a recurrent pattern. For example, at the morning and evening peak hour traffic volume is higher, which means the overall speed at this time period is lower. Similarly, traffic flow patterns are different on both weekdays and weekends. In case of weekdays, traffic volume is quite higher than the weekends. We apply the proposed method to predict queue lengths for north through traffic. Same methodology can be applied to predict queue lengths for all the other movements as well.

4. Case Study

4.1. Data Description

For this study, we collected adaptive traffic signal data from InSync between December 18, 2017 and February 14, 2018. We collected the data for the corridor of Alafaya Trail (SR-434) located in East Orlando, FL, from its Waterford lake intersection to Maculloch road intersection including 11 intersections (Fig. 3). InSync database provides two types of data: (i) Turning Movement Counts (TMC) - vehicle counts per phase and lane for every 15 minutes; (ii) Historical data with the details of each movement with the time, duration, queue, and wait time (refers to the wait time in seconds of the first car that was detected on the phase at the time logged) for each phase. In general, the historical data contain information regarding eight distinct movements North Left (NL), North Through (NT), South Left (SL), South Through (ST), East Left (EL), East Through (ET), West Left (WL) and West Through WT). Movements of pedestrians, bicycles or any non-motorized vehicle are considered as a separate phase.

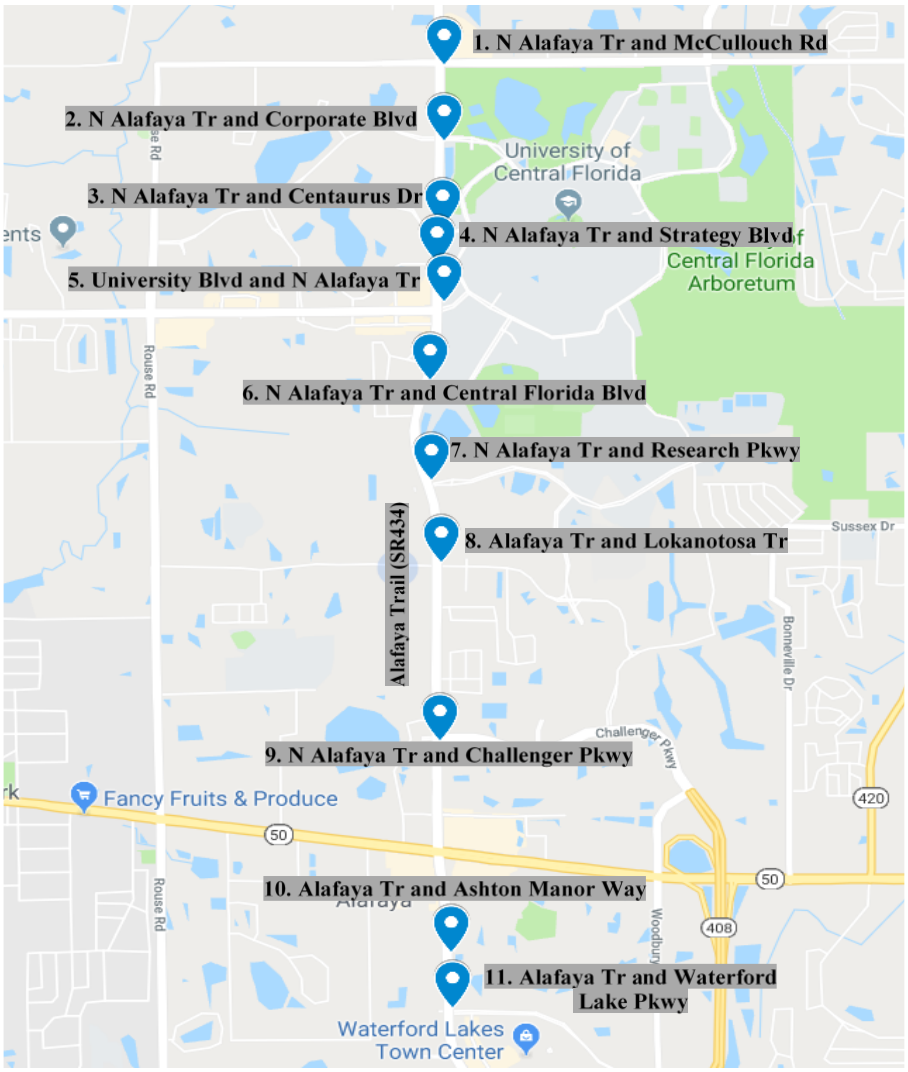
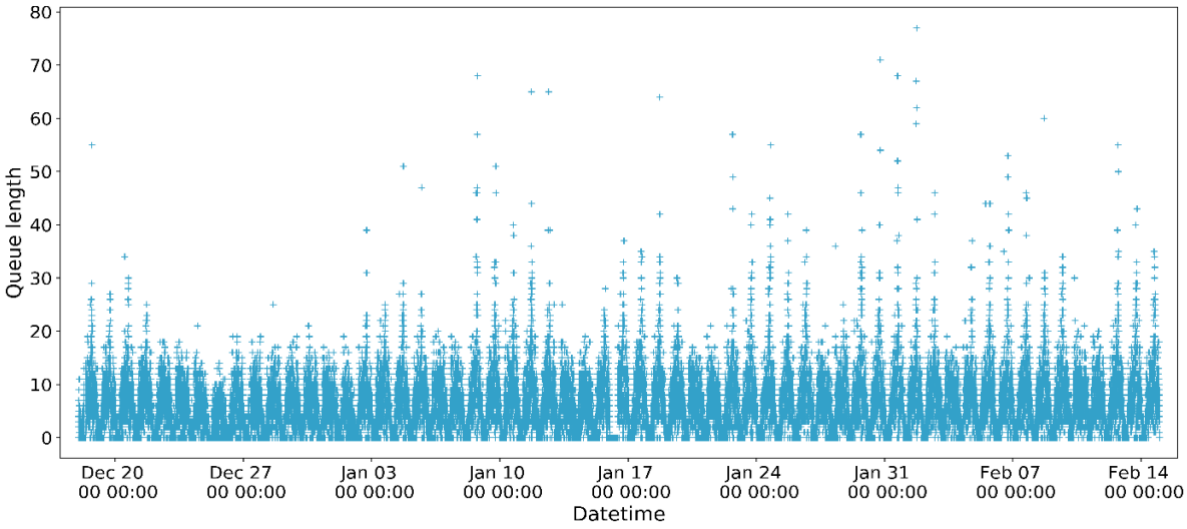


Fig. 3. Study Location (Google Map, 2018)

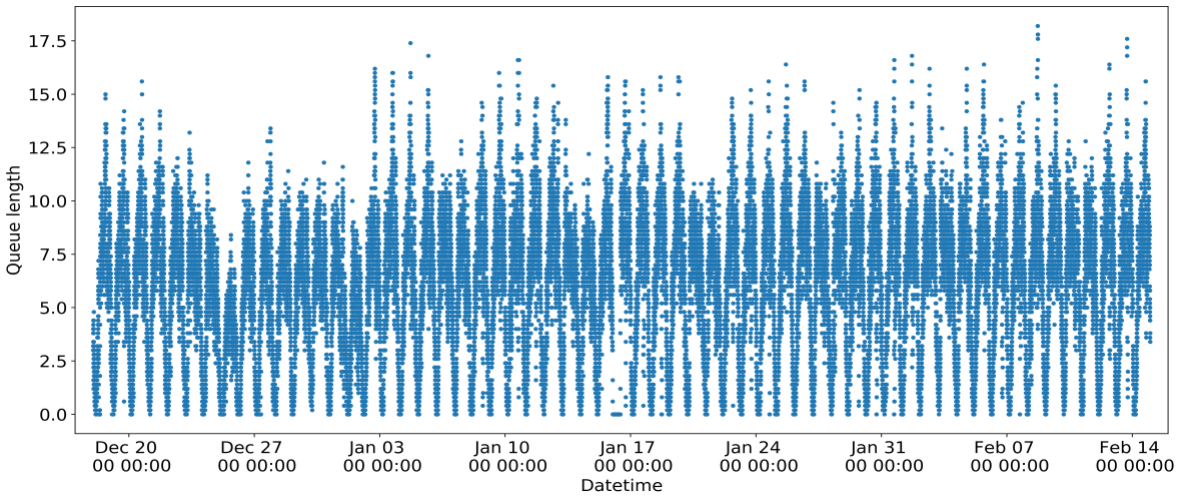
4.2. Data Preparation

In this study, we mainly focused on north through movements. We separated the data containing queue lengths (see Fig. 4 (a)) and wait times for the north through movement. The data collected from the phase history log contain multiple queue lengths for a given direction (north through) for a single cycle period which means that the same queue (north through direction) was cleared multiple times within a single cycle period. For our study corridor, the cycle period usually varies in between 120s to185s.

In general, the raw data collected from traffic sensors are subjected to errors. Several factors such as detector's malfunctioning, false encoding during data storage into the server, bad weather conditions etc. can cause errors. To understand the quality of the data we plotted the queue length with respect to time. Fig. 4(a) shows that few data points drastically deviated from the regular trends indicating that the collected data contains a few outliers which might cause poor fitting of the model. Hence, we apply two different approaches to remove the outliers and reduce noises in the data.



(a) Queue length with outliers



(b) Queue length without outliers

Fig. 4. Queue Length Variation over time for Alafaya-Mcculloch Intersection

First, we considered the maximum possible queue length detection by the detectors. InSync Adaptive traffic controller depends on the mounted video cameras to detect the number of vehicles and how long the vehicles have been waiting. In some cases, the detection system is fused with loop detectors to assist the queue detection. The detectors are placed at a certain distance from the stop line at the upstream of the intersection. The distance varies between 285 feet and 484 feet (*68*). Hence, maximum possible queue length detection by the detectors should be less than 35 (average vehicle length 14.5 feet). Considering this issue, we discarded the queue lengths greater than 40 from our analysis. Then we used interquartile range to remove the outliers. We chose a boundary in between 1.5 times the interquartile range and remove the queue lengths which fall outside this boundary. For prediction purpose, we chose the cycle length as 120 s and aggregated all the small queue lengths within a single cycle period. The objective is to predict the queue length for the next cycle (after 120 s). Finally, we applied a rolling average method over a window size

of 5 to reduce the noise. Fig. 4(b) shows the trends in queue length over time after cleaning the raw data.

4.3. Experiment Results

To predict the queue lengths for the next cycle time, we train the LSTM model with InSync data. We divide the data into two sets, first 80% of the data are used for training and next 20% of the data are used for validation. For selecting the hyperparameter for the deep LSTM NN model, we run the SMBO algorithm on different dataset corresponding to different intersections (1 to 9) and finally we obtain the optimal combination of hyperparameters which works best for each dataset.

While training the LSTM NN model, we do not pass the entire dataset, rather we divide the dataset into small batches. Hence, at each iteration the model learns the entire dataset in small batches and then move into next iteration and do the same. As shown in Table 1, we choose categorical distribution of batch size over {360, 720, 1440}. Based on our experiment, we find that the batch size does not influence the model accuracy, rather it influences the training time. In our case, we want to reduce the number of iterations to reach a stable solution. Hence, we choose the batch size of 1440. Table 2 shows the optimal parameters for the final LSTM-NN model.

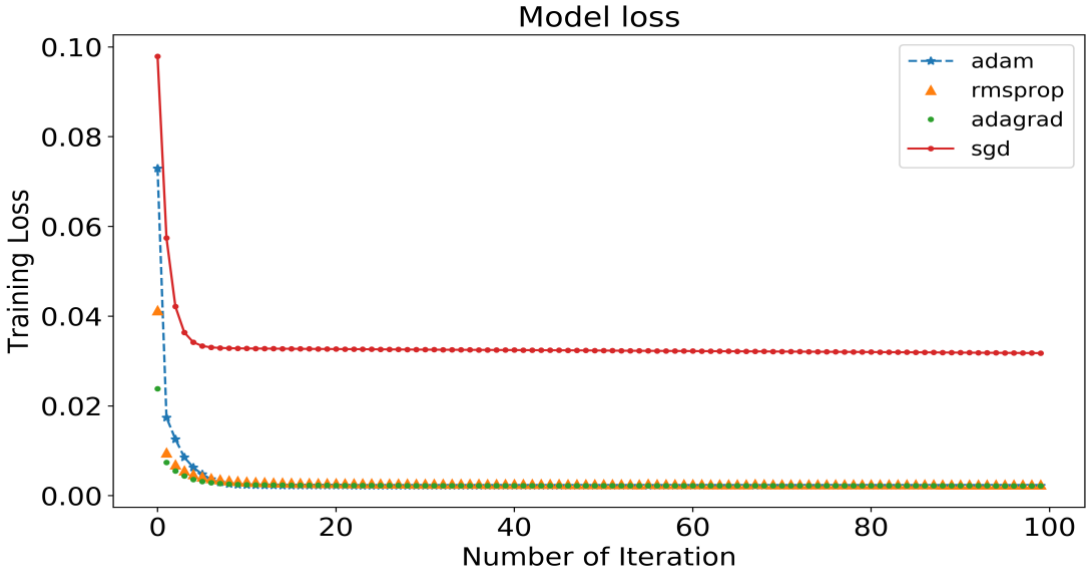


Fig. 5. Variation of Training Loss per Iteration for Different Optimizers (Batch Size =1440)

To check the convergence of the model, we observe the loss function value (mean squared error) and continue training the model unless there is no improvement in training loss for 20 iterations (Fig. 5 and Fig. 6). In most of the cases, after a certain number of iterations, the training loss is almost similar to the validation loss. We also use early stopping criteria to make sure the training loss is not higher than the validation loss. The early stopping method ensures termination of the training process if training loss value is greater than validation loss value. Thus, early stopping prevents model overfitting.

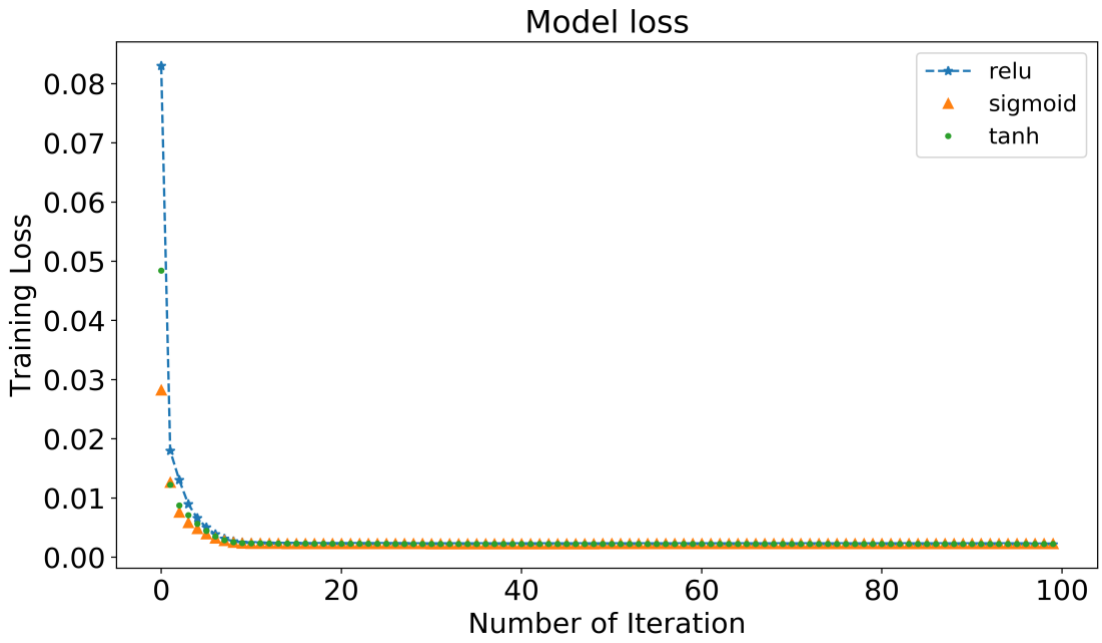


Fig. 6. Variation of Training Loss per Iteration for Different Activation Function (Batch Size =1440, Optimizer = adam)

Table 2 Hyperparameters for the best Performing Model

Number of Stacked LSTM Layers	Number of Hidden Units	Dropout	Activation Function	Optimizer
First	256	0.01655	relu	Adam
Second	128	0.00377	relu	

From the optimization result, we find that adam optimizer works better than rmsprop, adagrad and sgd optimizers. However as shown in Fig. 5, adam, rmsprop and adagrad have similar efficiency but adam optimizer converges faster than the others. Hence, it takes less time to train the model. Fig. 6 shows the training loss for different activation function. Both relu and tanh activation function work better, but if we choose sigmoid function the model starts overfitting at certain points before converging to the validation loss. Hence, we need to add large dropout at each layer to control the training process and it takes a long time to converge. The dropouts are added to control overfitting of the training set. But for our case the dropout value is so small, if we ignore these values (dropout =0), it does not affect the model performance. We have also applied the early stopping criteria to avoid overfitting. The model stops training when training loss is less than the validation loss. Fig. 7 shows the training and validation loss for the best model. We can see that the model converges after 70 iterations (epoch).

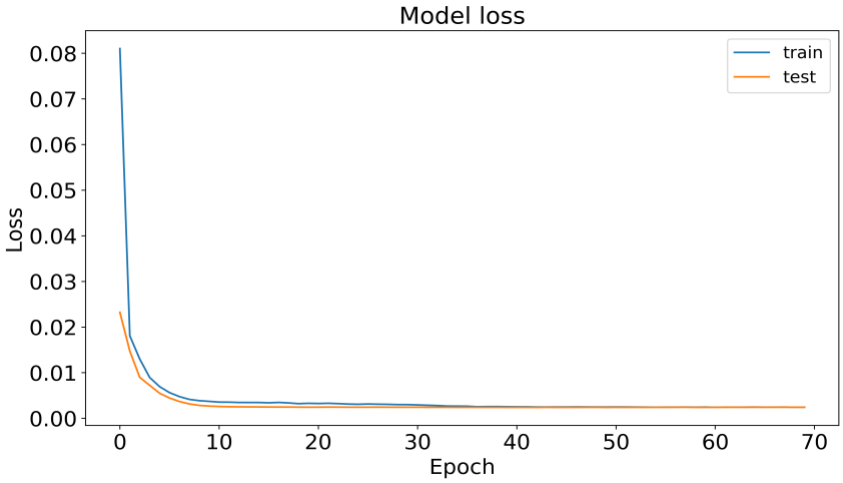


Fig. 7. Training and Validation Loss for the optimized model

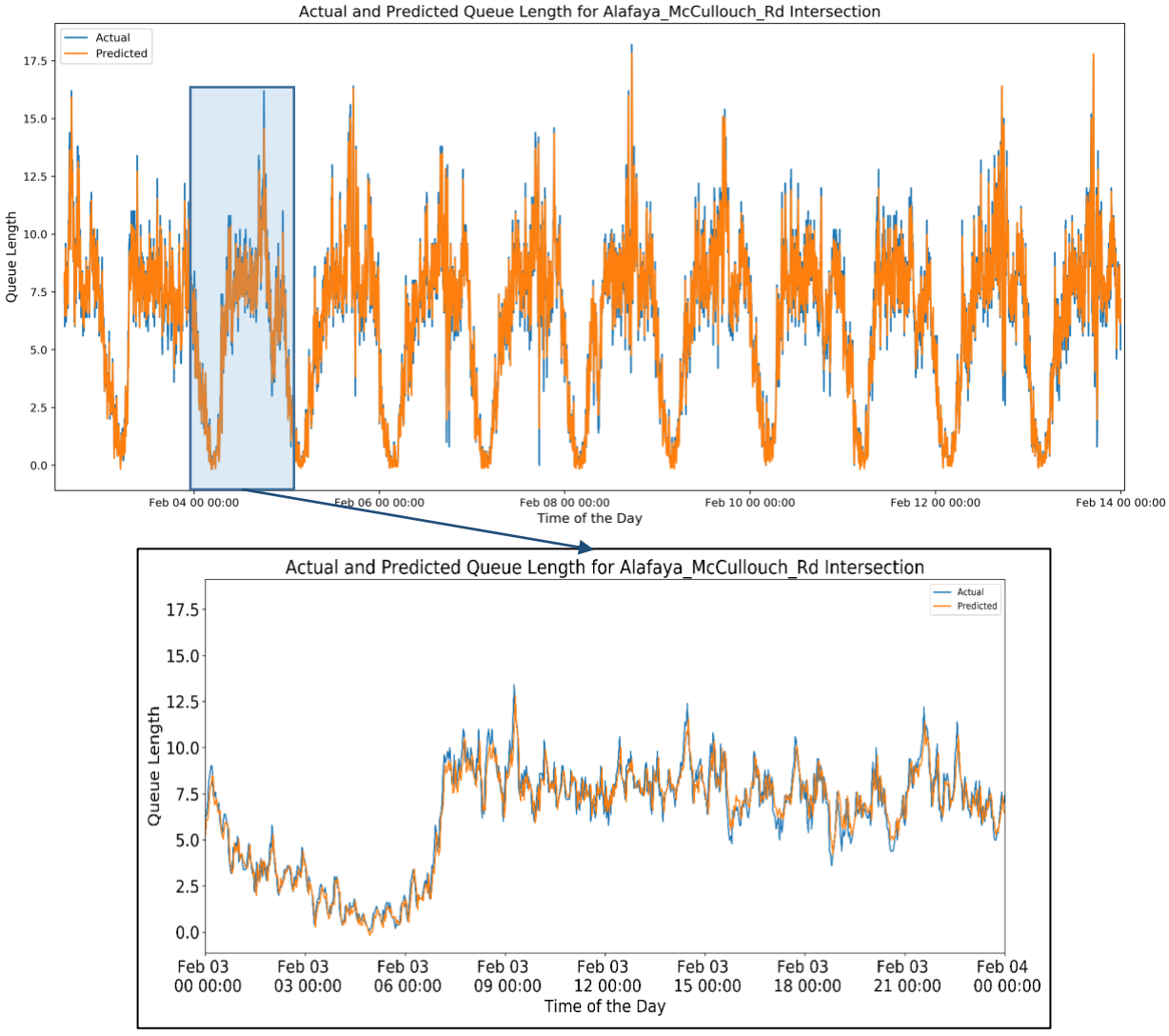


Fig. 8. Actual and Predicted Queue Length for Alafaya and McCullouch Road Intersection

Fig. 8 shows that the trained LSTM NN model performs very well to capture the variations of queue length over time. The difference between actual and predicted queue length is quite low. From Fig. 9, we observe that in most cases the difference between actual and predicted value for different intersection varies from 0.3 to 1.2. We have calculated Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) as performance measures to check the accuracy of the implemented model. Performance metrics are defined as,

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}} \quad (10)$$

$$\text{MAE} = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{n} \quad (11)$$

Fig. 10 shows that in most cases the RMSE values are less than 1. The maximum RMSE value is found for Alafaya Trail and Corporate Blvd intersection. While for each intersection, the MAE value is less than 1 as well.

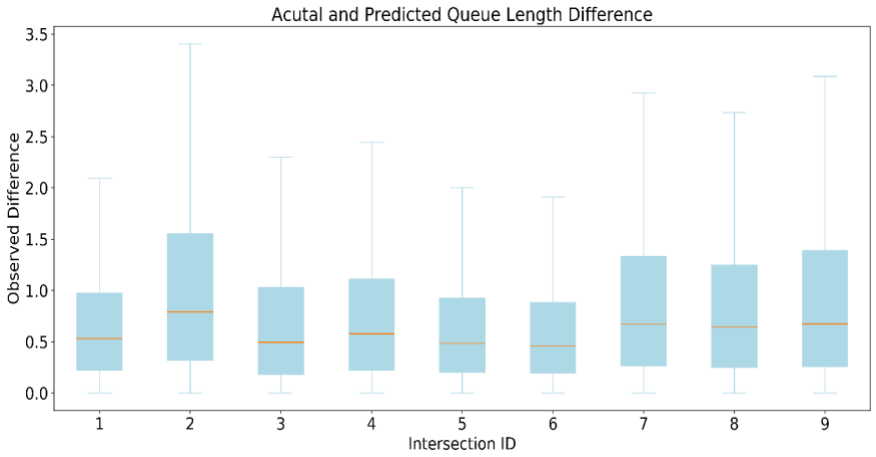


Fig. 9. Distribution of the Difference between Actual and Predicted Queue Length

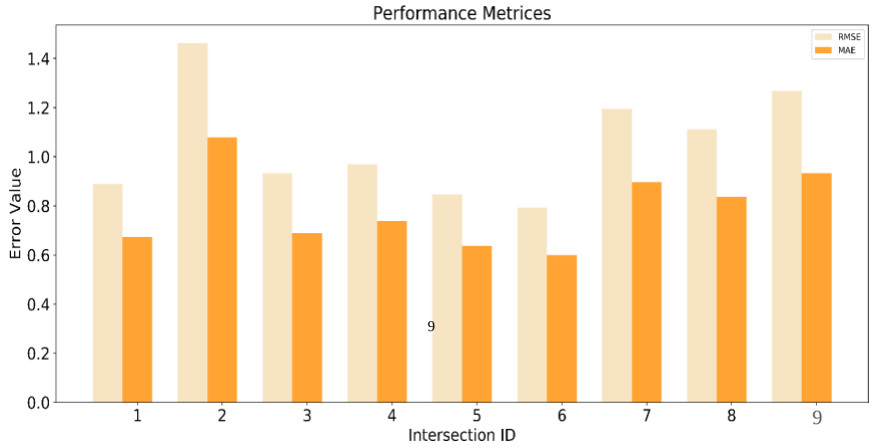
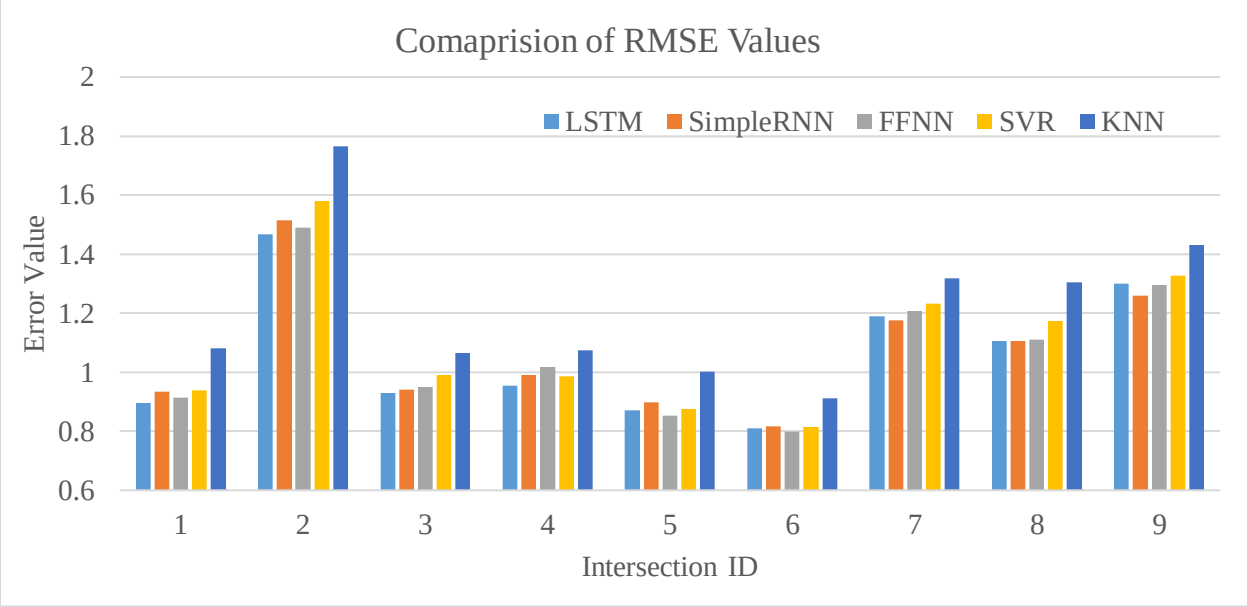


Fig. 10. Variation of Performance Metrics for Different Intersections

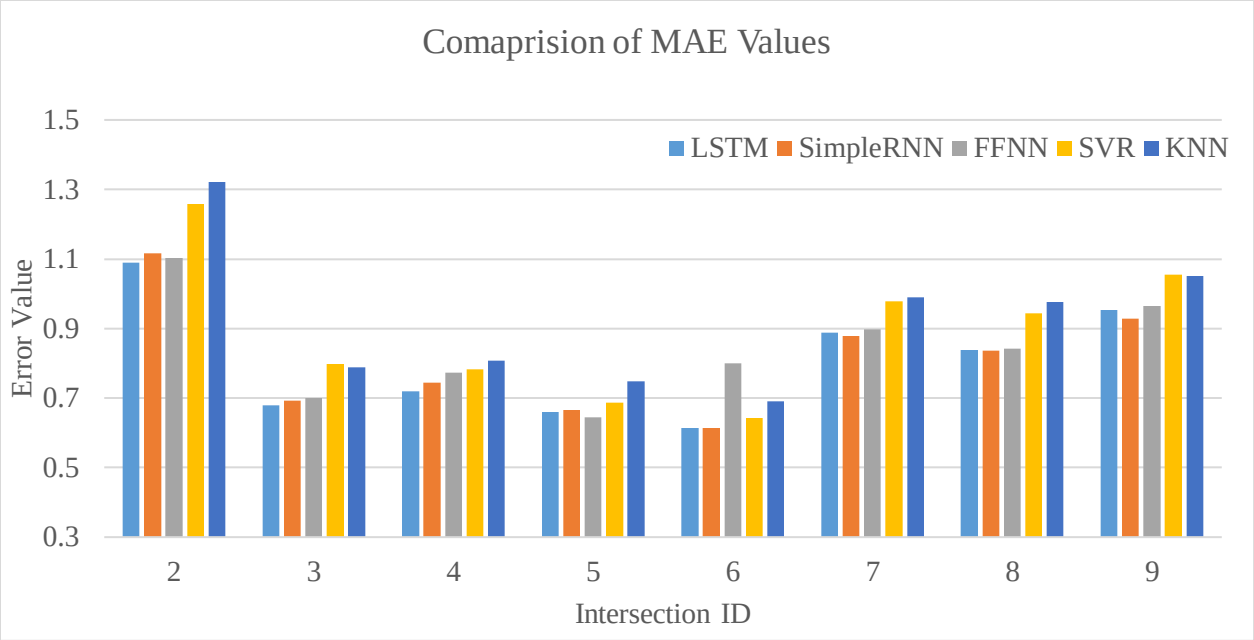
We also compare the LSTM NN model with other traditional models, such as support vector regression (SVR), K nearest neighbors (KNN) regression, and deep learning models such

as simple recurrent neural network (simpleRNN) and feedforward neural network (FFNN). Fig. 11 shows that, for all the intersections, the RMSE and MAE values for LSTM NN model are lower than the SVR and KNN (optimal number of neighbors =15). This suggests that traditional non neural network based approaches fail to capture the sharp variations of queue lengths. Therefore, we need a more robust model to predict real-time queue length variations. In case of deep neural network based method, we find that the models perform best if we use similar hyper parameters as the LSTM NN model and the performance is quite competitive with the proposed LSTM NN model. However, the LSTM NN model performs better in most of the cases, except for intersections 5 and 9, in which the Simple RNN and FFFNN model perform slightly better.

As such, the LSTM NN performs consistently better in most of the cases, if not all the cases. The LSTM NN model captures both the long-term and short-term dependencies. However, the other two models (simpleRNN and FFNN) rely on short-term dependencies of the traffic state variables. When predicting queue lengths, capturing long-term dependencies is valuable since traffic variation follows a regular pattern. Thus, the knowledge from historical trend is expected to improve the overall performance of a model. However, this can encounter another problem if the underlying data have some anomalies in the sequence. In such cases, model performance may deteriorate. The data from intersection 5 and intersection 9 may have some anomalies which slightly reduce the performance of the model when long-term dependencies are considered.



(a)



(b)

Fig. 11. Comparison of Performance Metrics for Different Models

5. Conclusions

In advanced traffic control systems, accurate queue length prediction is a critical component to ensure efficient traffic operations at an intersection. With the advancement in technologies, several techniques (computer vision, detectors, V2X communication) have been introduced to accurately gather information at the intersection for optimal traffic control. However, the system will not operate in an optimal level unless the future traffic state (queue length, queue volume etc.) has been predicted accurately. In this study, we have developed a data-driven method to predict queue lengths in the next cycle from real-time traffic data. Assuming a connected corridor, we have implemented a deep LSTM-NN model to predict the queue length for the next cycle. The experiment results show that a deep learning based method performs very well to capture the time-dependent patterns of traffic signal queues. Moreover, we have adapted a new approach for hyperparameter optimization for deep learning method using SMBO technique. This approach reduces the complexity and effort required to select best possible hyperparameters and avoid overfitting. We also compare our model with other data-driven approaches, showing that the proposed LSTM NN model performs better than the existing (including traditional and deep learning based) approaches.

One of the major benefits of the proposed deep learning model is that it can be implemented in real time and can be updated based on real-time signal data. Though we have used only the queue length and wait time as input features, we can add more features related to vehicular traffic states (traffic flow, average travel time or speed) merging data from multiple sources to provide a more complete picture of signal states for a better prediction.

This study is an initial step towards development of data-driven traffic control and optimization methods using real-time traffic data. However, due to the lack of data sources we

could not collect the information on vehicular movements for different direction at a high resolution (30s to 60s). If we can include such information, the performance of the model may further improve.

Future studies should conduct experiments for a complete intersection considering the queue length for each lane. Different computer vision based approaches to collect high resolution data for vehicular movements can also be considered. An optimization technique for the adaptive traffic control system based on the predicted queue lengths can be developed. Although we considered fixed cycle times only, to implement in practice, the model needs to be more flexible so that it can predict the queue length for variable cycle times. Furthermore, our method can be extended developing an algorithm that can determine the next cycle length based on current traffic state and delay.

Acknowledgement

The authors acknowledge Prof. Mohamed Abdel-Aty and Yaobang Yang for providing us the InSync datasets for training the algorithms.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Schrank., D., B. Eisele., T. Lomax., and J. Bak. 2015 Urban Mobility Scorecard. *Texas A&M Transportation Institue*, Vol. 39, No. August, 2015, p. 5. https://doi.org/DTRT06-G-0044.
2. Smith, S. F., G. J. Barlow, X.-F. Xie, and Z. B. Rubinstein. Smart Urban Signal Networks: Initial Application of the SURTRAC Adaptive Traffic Signal Control System. *Icaps*, 2013, pp. 434–442.
3. Chandra, R., and C. Gregory. InSync Adaptive Traffic Signal Technology : Real-Time Artificial Intelligence Delivering Real-World Results. *InSync White Paper*, No. March, 2012, p. 26.
4. Liu, H. X., X. Wu, W. Ma, and H. Hu. Real-Time Queue Length Estimation for Congested Signalized Intersections. *Transportation Research Part C: Emerging Technologies*, Vol. 17, No. 4, 2009, pp. 412–427. https://doi.org/10.1016/j.trc.2009.02.003.
5. Feng, Y., K. L. Head, S. Khoshmagham, and M. Zamanipour. A Real-Time Adaptive Signal Control in a Connected Vehicle Environment. *Transportation Research Part C: Emerging Technologies*, Vol. 55, 2015, pp. 460–473. https://doi.org/10.1016/j.trc.2015.01.007.
6. Jing, P., H. Huang, and L. Chen. An Adaptive Traffic Signal Control in a Connected Vehicle Environment: A Systematic Review. *Information (Switzerland)*, Vol. 8, No. 3, 2017. https://doi.org/10.3390/info8030101.
7. Mirchandani, P., and L. Head. RHODES: A Real-Time Traffic Signal Control System. *Architecture, Algorithms and Analysis*, 1998, pp. 1–15.

8. Miyato, T., A. M. Dai, and I. Goodfellow. Adversarial Training Methods for Semi-Supervised Text Classification. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019, pp. 1–11.
9. Rosen-Zvi, M., T. Griffiths, M. Steyvers, and P. Smyth. The Author-Topic Model for Authors and Documents. *arXiv preprint arXiv:1207.4169*, 2012.
10. Yilmaz, A., O. Javed, and M. Shah. Object Tracking: A Survey. *ACM Computing Surveys*, Vol. 38, No. 4, 2006. https://doi.org/10.1145/1177352.1177355.
11. Finn, C., I. Goodfellow, and S. Levine. Unsupervised Learning for Physical Interaction through Video Prediction. *Advances in Neural Information Processing Systems*, No. Nips, 2016, pp. 64–72.
12. Bi, J., H. Yuan, and M. Zhou. Temporal Prediction of Multiapplication Consolidated Workloads in Distributed Clouds. *IEEE Transactions on Automation Science and Engineering*, Vol. 16, No. 4, 2019, pp. 1763–1773. https://doi.org/10.1109/TASE.2019.2895801.
13. Drucker, H., C. J. C. Surges, L. Kaufman, A. Smola, and V. Vapnik. Support Vector Regression Machines. *Advances in Neural Information Processing Systems*, Vol. 1, 1997, pp. 155–161.
14. Cortes, C., and V. Vapnik. SUPPORT-VECTOR NETWORKS 1 Introduction. *Machine Learning*, Vol. 20, No. 3, 1995, pp. 273–297.
15. Altman, N. S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, Vol. 46, No. 3, 1992, pp. 175–185.
16. Werbos, P. J. Backpropagation through Time: What It Does and How to Do It. *Proceedings of the IEEE*, Vol. 78, No. 10, 1990, pp. 1550–1560. https://doi.org/10.1109/5.58337.
17. Hopfield, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 79, No. April, 1982, pp. 2254–2558. https://doi.org/10.1201/9780429500459.
18. Geurts, M., G. E. P. Box, and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 1977.
19. Yu, B., X. Song, F. Guan, Z. Yang, and B. Yao. K-Nearest Neighbor Model for Multiple-Time-Step Prediction of Short-Term Traffic Condition. *Journal of Transportation Engineering*, Vol. 142, No. 6, 2016, p. 04016018. https://doi.org/10.1061/(ASCE)TE.1943-5436.0000816.
20. Deshpande, M., and P. R. Bajaj. Performance Analysis of Support Vector Machine for Traffic Flow Prediction. *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 126–129. https://doi.org/10.1109/ICGTSPICC.2016.7955283.
21. Wu, C., C. Wei, D. Su, M. Chang, and J. Ho. Travel Time Prediction with Support Vector Regression. *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, Vol. 2, 2004, pp. 1438–1442. https://doi.org/10.1109/ITSC.2003.1252721.
22. Billings, D., and Y. Jiann-Shiou. Application of the ARIMA Models to Urban Roadway Travel Time Prediction-A Case Study. Systems, Man and Cybernetics, 2006. SMC'06. *IEEE International Conference on*, 2006, pp. 2529–2534.
23. Lee, Y. L. Y. Freeway Travel Time Forecast Using Artifical Neural Networks with

Cluster Method. *2009 12th International Conference on Information Fusion*, 2009, pp. 1331–1338.
24. Yasdi, R. Prediction of Road Traffic Using a Neural Network Approach. *Neural Computing and Applications*, Vol. 8, No. 2, 1999, pp. 135–142. https://doi.org/10.1007/s005210050015.
25. Hong, W. C. Application of Seasonal SVR with Chaotic Immune Algorithm in Traffic Flow Forecasting. *Neural Computing and Applications*, Vol. 21, No. 3, 2012, pp. 583–593. https://doi.org/10.1007/s00521-010-0456-7.
26. Hodge, V. J., R. Krishnan, J. Austin, J. Polak, and T. Jackson. Short-Term Prediction of Traffic Flow Using a Binary Neural Network. *Neural Computing and Applications*, Vol. 25, No. 7–8, 2014, pp. 1639–1655. https://doi.org/10.1007/s00521-014-1646-5.
27. 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. https://doi.org/10.1016/j.trc.2017.02.024.
28. Ai, Y., Z. Li, M. Gan, Y. Zhang, D. Yu, W. Chen, and Y. Ju. A Deep Learning Approach on Short-Term Spatiotemporal Distribution Forecasting of Dockless Bike-Sharing System. *Neural Computing and Applications*, Vol. 31, No. 5, 2019, pp. 1665–1677. https://doi.org/10.1007/s00521-018-3470-9.
29. Shukla, S., K. Balachandran, and V. S. Sumitha. A Framework for Smart Transportation Using Big Data. *Proceedings of 2016 International Conference on ICT in Business, Industry, and Government, ICTBIG 2016*, 2017, pp. 1–3. https://doi.org/10.1109/ICTBIG.2016.7892720.
30. Ma, X., Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long Short-Term Memory Neural Network for Traffic Speed Prediction Using Remote Microwave Sensor Data. *Transportation Research Part C: Emerging Technologies*, Vol. 54, 2015, pp. 187–197. https://doi.org/10.1016/j.trc.2015.03.014.
31. Yu, H., Z. Wu, S. Wang, Y. Wang, and X. Ma. Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks. *Sensors (Switzerland)*, Vol. 17, No. 7, 2017, pp. 1–16. https://doi.org/10.3390/s17071501.
32. Rahman, R., and S. Hasan. Short-Term Traffic Speed Prediction for Freeways During Hurricane Evacuation : A Deep Learning Approach. 2018, pp. 1291–1296. https://doi.org/10.1109/ITSC.2018.8569443.
33. Newell, G. F. Approximation Methods for Queues with Application to the Fixed-Cycle Traffic Light. *Society for Industrial and Applied Mathematics*, Vol. 7, No. 2, 1965, pp. 223–240.
34. Chang, T. H., and J. T. Lin. Optimal Signal Timing for an Oversaturated Intersection. *Transportation Research Part B: Methodological*, Vol. 34, No. 6, 2000, pp. 471–491. https://doi.org/10.1016/S0191-2615(99)00034-X.
35. Mirchandani, P. B., and N. Zou. Queuing Models for Analysis of Traffic Adaptive Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 8, No. 1, 2007, pp. 50–59. https://doi.org/10.1109/TITS.2006.888619.
36. Balke, K. N., H. Charara, and R. Parker. Development of a Traffic Signal Performance Measurement System (TSPMS). Vol. 7, No. May 2005, 2005, p. 83.
37. Webster, F. V. Traffic Signal Settings. *Road Research Technical paper 39*, 1957.
38. Robertson, D. *TRANSYT: A Traffic Network Study Tool*. 1969.
39. May, A. D. *Traffic Flow Theory- the Traffic Engineers Challenge.* 1975.

40. Sharma, A., D. M. Bullock, J. A. Bonneson, A. Sharma, D. M. Bullock, and J. A. Bonneson. Input-Output and Hybrid Techniques for Real- Time Prediction of Delay and Maximum Queue Length at Signalized Intersections Delay and Maximum Queue Length at Signalized Intersections. *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2035, 2007, pp. 69–80. https://doi.org/10.3141/2035-08.
41. Vigos, G., M. Papageorgiou, and Y. Wang. Real-Time Estimation of Vehicle-Count within Signalized Links. *Transportation Research Part C: Emerging Technologies*, Vol. 16, No. 1, 2008, pp. 18–35. https://doi.org/10.1016/j.trc.2007.06.002.
42. Stephanopoulos, G., P. G. Michalopoulos, and G. Stephanopoulos. Modelling and Analysis of Traffic Queue Dynamics at Signalized Intersections. *Transportation Research Part A: General*, Vol. 13, No. 5, 1979, pp. 295–307. https://doi.org/10.1016/0191-2607(79)90028-1.
43. Lighthill, M. J., and G. B. Whitham. On Kinematic Waves II. A Theory of Traffic Flow on Long Crowded Roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, Vol. 229, No. 1178, 1955, pp. 317–345. https://doi.org/10.1098/rspa.1955.0089.
44. Richards, P. I. Shock Waves on the Highway. *Operations Research*, Vol. 4, No. 1, 1956, pp. 42–51. https://doi.org/10.1287/opre.4.1.42.
45. Smaglik, E., A. Sharma, D. Bullock, J. Sturdevant, and G. Duncan. Event-Based Data Collection for Generating Actuated Controller Performance Measures. *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2035, No. 2035, 2007, pp. 97–106. https://doi.org/10.3141/2035-11.
46. An, C., Y. J. Wu, J. Xia, and W. Huang. Real-Time Queue Length Estimation Using Event-Based Advance Detector Data. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, Vol. 0, No. 0, 2017, pp. 1–14. https://doi.org/10.1080/15472450.2017.1299011.
47. Jeff Ban, X., P. Hao, and Z. Sun. Real Time Queue Length Estimation for Signalized Intersections Using Travel Times from Mobile Sensors. *Transportation Research Part C: Emerging Technologies*, Vol. 19, No. 6, 2011, pp. 1133–1156. https://doi.org/10.1016/j.trc.2011.01.002.
48. Hao, P., and X. Ban. Long Queue Estimation for Signalized Intersections Using Mobile Data. *Transportation Research Part B: Methodological*, Vol. 82, 2015, pp. 54–73. https://doi.org/10.1016/j.trb.2015.10.002.
49. Comert, G. Simple Analytical Models for Estimating the Queue Lengths from Probe Vehicles at Traffic Signals. *Transportation Research Part B: Methodological*, Vol. 55, 2013, pp. 59–74. https://doi.org/10.1016/j.trb.2013.05.001.
50. Tiaprasert, K., Y. Zhang, X. B. Wang, and X. Zeng. Queue Length Estimation Using Connected Vehicle Technology for Adaptive Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 4, 2015, pp. 2129–2140. https://doi.org/10.1109/TITS.2015.2401007.
51. Emami, A., M. Sarvi, and S. Asadi Bagloee. A Neural Network Algorithm for Queue Length Estimation Based on the Concept of K-Leader Connected Vehicles. *Journal of Modern Transportation*, Vol. 27, No. 4, 2019, pp. 341–354. https://doi.org/10.1007/s40534-019-00200-y.
52. Gao, K., F. Han, P. Dong, N. Xiong, and R. Du. Connected Vehicle as a Mobile Sensor

for Real Time Queue Length at Signalized Intersections. *Sensors (Switzerland)*, Vol. 19, No. 9, 2019, pp. 1–22. https://doi.org/10.3390/s19092059.

53. Chang, G. L., and C. C. Su. Predicting Intersection Queue with Neural Network Models. *Transportation Research Part C*, Vol. 3, No. 3, 1995, pp. 175–191. https://doi.org/10.1016/0968-090X(95)00005-4.
54. Lee, S., K. Xie, D. Ngoduy, and M. Keyvan-Ekbatani. An Advanced Deep Learning Approach to Real-Time Estimation of Lane-Based Queue Lengths at a Signalized Junction. *Transportation Research Part C: Emerging Technologies*, Vol. 109, 2019, pp. 117–136. https://doi.org/10.1016/j.trc.2019.10.011.
55. Gan, S., S. Liang, K. Li, J. Deng, and T. Cheng. Trajectory Length Prediction for Intelligent Traffic Signaling: A Data-Driven Approach. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 19, No. 2, 2018, pp. 426–435. https://doi.org/10.1109/TITS.2017.2700209.
56. Wang, J., and T. Kumbasar. Parameter Optimization of Interval Type-2 Fuzzy Neural Networks Based on PSO and BBBC Methods. *IEEE/CAA Journal of Automatica Sinica*, Vol. 6, No. 1, 2019, pp. 247–257. https://doi.org/10.1109/JAS.2019.1911348.
57. Gao, S., M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang. Dendritic Neuron Model with Effective Learning Algorithms for Classification, Approximation, and Prediction. *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 30, No. 2, 2019, pp. 601–614. https://doi.org/10.1109/TNNLS.2018.2846646.
58. Zeng, J., S. Yu, Y. Qian, and X. Feng. Expressway Traffic Flow Model Study Based on Different Traffic Rules. *IEEE/CAA Journal of Automatica Sinica*, Vol. 5, No. 6, 2018, pp. 1099–1103. https://doi.org/10.1109/JAS.2017.7510469.
59. Lipton, Z. C., J. Berkowitz, and C. Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning ArXiv : 1506 . 00019v4 [Cs . LG] 17 Oct 2015. 2015, pp. 1–38.
60. Hochreiter, S., and J. Urgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
61. Géron, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media.
62. Wang, G., J. Qiao, J. Bi, W. Li, and M. Zhou. TL-GDBN: Growing Deep Belief Network with Transfer Learning. *IEEE Transactions on Automation Science and Engineering*, Vol. 16, No. 2, 2019, pp. 874–885. https://doi.org/10.1109/TASE.2018.2865663.
63. Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 2546–2554. https://doi.org/2012arXiv1206.2944S.
64. Bergstra, J., D. Yamins, and D. D. Cox. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. *12th PYTHON IN SCIENCE CONF. (SCIPY 2013)*, No. Scipy, 2013, pp. 13–20. https://doi.org/10.1088/1749-4699/8/1/014008.
65. Hutter, F., H. H. Hoos, and K. Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6683 LNCS, 2011, pp. 507–523. https://doi.org/10.1007/978-3-642-25566-3_40.
66. Jones, D. R. A Taxonomy of Global Optimization Methods Based on Response Surfaces.

Journal of Global Optimization, Vol. 21, 2001, pp. 345–383. https://doi.org/10.1023/A:1012771025575.
67. Rahman, R. Applications of Deep Learning Models for Traffic Prediction Problems. 2019.
68. Traffic, T., and D. Manual. *CHAPTER 8 TRAFFIC SIGNAL DESIGN*. 2016.