Proceedings of the 7th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'21)
~~Prague, Czech Republic~~ Virtual Conference - July, 2021
Paper No. MVML 102
DOI: 10.11159/mvml21.102

Short-Term Traffic Forecasting Using Deep Learning

Iren Valova[1], Natacha Gueorguieva[2], Sandeep Smudidonga[2],
[1]Computer and Information Science Department, University of Massachusetts Dartmouth
285 Old Westport Rd, Dartmouth, MA, USA
Iren.Valova@umassd.edu
[2]Department of Computer Science, College of Staten Island/City University of New York
2800 Victory Blvd, NY, USA
Natacha.Gueorguieva@csi.cuny.edu; Sandeep.Smudidonga@cix.csi.cuny.edu

Abstract - Forecasting is one of the key applications of machine learning. The task of forecasting becomes complex when there are spatiotemporal dependencies in the data generating process. Prediction of congestion ahead of time is a very important aspect of transportation system management. Traffic congestion on a road network has a temporal component due to daily and weekly variation in human travel, and also a spatial component due to the connected nature of the road network and traffic flow. Furthermore, the spatial component of traffic congestion is certainly not Euclidean due to directionality of road network, which is not an undirected graph. Congestion prediction falls into the realm of time series data analysis methods which can be mapped onto a neural network-based methods for sequence prediction. In this research we propose Convolutional Long Short Term Memory (CLSTM) which incorporates spatial and temporary information into the forecasting process. To validate the efficiency of the proposed method, the performance is compared with various deep learning architectures of Gated Recurrent Unit (GRU), Long Short Term Memory (LSTM), and baseline methods such as Vector Autoregression (VAR) and historical average. Experiments include the above topologies with varying parameters as number of units per layer, number of layers, optimizers, learning rate and lengths of sequence input. Prediction comparison is demonstrated with tables and graphical representations.

Keywords**:** convolutional neural networks, long short term memory; traffic flow prediction; gated units

1. Introduction

Traffic congestion results in lost time for travellers, extra fuel consumption, and low speed and idling vehicles cause air pollution. Lack of proper information on congestion can also result in increased delays, accidents and further congestion due to uninformed travel choices [1]. The impact of congestion can be mitigated using advanced information provision to navigation systems and also to highway information message signs. Congestion prediction can also provide insight into analysis of congestion propagation and identification of traffic flow bottlenecks, which can help in strategizing and formulating solutions.

Traffic forecasting based on knowledge-driven methods mainly use queuing theory to simulate user behaviour [2] while data-driven approach for traffic congestion prediction are centred on time series-based models. Traditional time series methods as Auto-Regressive Integrated Moving Average (ARIMA) [3], Kalman filtering [4, 5], principal component analysis [6], mixture models and Markov chain [7, 8] may be applicable to predicting recurrent congestion as a process that is repetitive and periodic. Non-stationary effect however may not be easily captured through time series methods as well as spatial effects of congestion and they cannot mine the deep relationships between data.

Though, in literature traditional time series and machine learning models discussed above have been used in solving traffic congestion, the highly nonlinear nature of traffic data justifies the use of neural network-based approaches as these methods can theoretically recreate any nonlinear function. [9]. Recently proposed Neural Network (NN) models with Deep Learning (DL) with their abilities to mine big data and discover internal structure and potential characteristics are successfully implemented for solving traffic flow prediction problem.

Huang [10] proposed a deep architecture for traffic low prediction with deep belief networks (DBN) and multitask learning, authors in [11] use a stacked autoencoder (SAE) model to learn generic traffic flow features. Recurrent neural network (RNN) combined with Restricted Boltzmann Machines (RBM) are used to extract features from high dimensional traffic data in order to forecast network travel time [12].

Two architectures that have been streamlined for time series modelling are Gated Recurrent Unit (GRU) [14] and Long Short Term Memory (LSTM) [13]. LSTM architecture uses three gates to control the cell state update – forget, input and output. The GRU combines forget and input into an update gate [15]. GRU is a kind of RNN, but its cell structure is simpler than LSTM cell structure. Because GRU simplifies the structure of memory unit, some important information may be ignored in the prediction process, and the prediction accuracy is usually reduced. Similar to GRU, LSTM is also an improved RNN, which can retain the influence of data for a longer time and improve the gradient disappearance for RNN which defines LSTM models suitable for time series prediction [16, 17, 18].

Convolutional neural networks and LSTM combined with RNNs are also used to analyse travel time forecast [19] and precipitation casting to predict the future rainfall intensity in a local region over a relatively short period of time [20]. Proposed Diffusion Convolutional Gated Recurrent Unit (DCGRU) for prediction traffic congestion presents spatial dependency as a diffusion process on the traffic network graph and implements RNN to model the temporal dependencies [21]. Different time series models, regularized time series model and LSTM are compared in travel time prediction [22]. In [23] LSTM network is augmented by autoencoders to feed spatial information as an additional input to traffic flow prediction.

The objective of this research is to predict travel time on highways over a short period of time in future. The problem is presented as a regression model involving congestion information from different sections on a highway over several days. We introduce a deep learning architecture CLSTM which integrates CNN and LSTM and uses sequences of past observations to forecast the future. Various experiments using CLSTM and several deep learning neural network architectures were explored and evaluated for their prediction accuracy with implementation of different optimizers, adaptive learning rate and sequence length. Evaluation of predictive accuracy across different models is using Mean Average Error (MAE), Root Mean Squared error (RMSE), Mean Relative Error (MRE) and Prediction Accuracy (ACC) for 15, 30 and 60 minutes in the future.

2. Deep Neural Networks for Sequence Modelling

Recurrent Neural Networks (RNNs) are a family of neural networks for processing a sequence of values $x(1), x(2), ..., x(\tau)$ and some of them can also process sequences of variable length [25]. Recurrent computations in RNN which involve mapping inputs and parameters are formalized with computational graphs because of their repetitive configuration which corresponds to a chain of NN events. Unfolding these graphs result in sharing of parameters across the RNN structure.

The recurrence computations of unfolded computational graph are based on the classical form of a dynamical system driven by an external signal $\boldsymbol{x}^{(t)}$ given by Eq.1:

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \theta) \tag{1}$$

where $\boldsymbol{s}^{(t)}$ is the state of the system at time *t* and θ is the set of shared parameters. Many RNNs are using (1) or similar to it equation in order to define the values of their hidden units $\boldsymbol{h}$. Therefore (1) takes the following form:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \theta) \tag{2}$$

When the recurrent network is trained to perform a task that requires predicting the future from the past, the network typically learns to use $\boldsymbol{h}^{(t)}$ with some losses since it maps an arbitrary length sequence $(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, ..., \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)})$ to a fixed length vector $\boldsymbol{h}^{(t)}$.

Gated RNNs as LSTM and GRU are the most successful sequence models in solving wide range of practical problems. They are based on the idea of creating paths through time that have derivatives which neither vanish nor explode. While standard RNN computes hidden layer at next time step directly using the input vector $\boldsymbol{x}$, hidden state $\boldsymbol{h}$, weight matrices $\boldsymbol{W}$ and vector $\boldsymbol{b}$ as shown in Eq.3:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{W}_{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{hx}\boldsymbol{x}^{(t)} + \boldsymbol{b}) \tag{3}$$

GRU first computes an *update gate* (another layer) which controls what parts of hidden state are updated vs. preserved:

$$\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{W}_u\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u\boldsymbol{x}^{(t)} + \boldsymbol{b}_u) \tag{4}$$

The next computations are on *reset gate* which determines what parts of the previous hidden state will be used to compute the new content:

$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r \boldsymbol{x}^{(t)} + \boldsymbol{b}_r) \tag{5}$$

New hidden state content resets the gate selected useful parts of previous content:

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_h (\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h \boldsymbol{x}^{(t)} + \boldsymbol{b}_h) \tag{6}$$

Update simultaneously controls what is kept from previous hidden state and what is updated to the new hidden state:

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)} \tag{7}$$

where $\boldsymbol{x}^{(t)}$ and $\boldsymbol{h}^{(t)}$ are the input and output vectors, $\tilde{\boldsymbol{h}}^{(t)}$ - candidate activation vector, $\boldsymbol{u}^{(t)}$ and $\boldsymbol{r}^{(t)}$ are the update and reset vectors; $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ are weight matrices and bias vector parameters which need to be learned during training; σ is sigmoid activation function and $\circ$ denotes the Hadamard product.

LSTM are proposed as a solution to the vanishing gradients problem [14, 15]. On step t, there is a hidden state and a cell state presented by vectors with length *n*. The cell stores long-term information and the LSTM can erase, write and read information from the cell. The selection of which information is erased/written/read is controlled by the following three dynamic gates with length *n*: (a) *forget gate* ($\boldsymbol{f}^{(t)}$) controls what is kept vs forgotten, from previous cell state; (b) *input gate* ($\boldsymbol{i}^{(t)}$) controls what parts of the new cell content are written to cell; (c) *output gate* ($\boldsymbol{o}^{(t)}$) controls what parts of cell are output to hidden state. On each time step, each element of the gates can be open (1), closed (0), or somewhere in-between and their value is computed based on the current context. *New cell content* ($\tilde{\boldsymbol{c}}^{(t)}$) determines the new content to be written to the cell; *cell state* ($\boldsymbol{c}^{(t)}$) comes with two options: erase ("forget") some content from last cell state, and write ("input") some new cell content; *hidden state* ($\boldsymbol{h}^{(t)}$) reads ("output") some content from the cell. Gates are applied using the Hadamard product. Equations (8) represent the forward pass of LSTM [15].

$$\begin{aligned}
\boldsymbol{f}^{(t)} &= \sigma(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f) \\
\boldsymbol{i}^{(t)} &= \sigma(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i) \\
\boldsymbol{o}^{(t)} &= \sigma(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{c}}^{(t)} &= \tanh(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c) \\
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}
\end{aligned} \tag{8}$$

where $\boldsymbol{x}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{i}^{(t)}$ and $\boldsymbol{o}^{(t)}$ are respectively input vector to LSTM unit, forget gate's activation vector, input/update gate's activation vector and output gate's activation vector; $\boldsymbol{h}^{(t)}$, $\tilde{\boldsymbol{c}}^{(t)}$ and $\boldsymbol{c}^{(t)}$ are in turn hidden state vector, cell input activation vector and cell state vector.

2.1. LSTM Combined with Convolutional NN

The problem of traffic congestion on highways is important, despite the regularity with which it happens. Let us denote the congestion observations as $\boldsymbol{X} \in \Re^{NxP}$ where N represents the distance between the nodes, *P* is the dimension of congestion metrics (speed, flow, etc.) and $\Re$ denotes the domain of the observed features. The spatial region defined above can be considered as N x P grid to each cell of which we can add the number of measurements *M* which differ over time. Therefore, a specific observation at any time can be represented by a tensor $\aleph \in \Re^{MxNxP}$. Considering the last notation, we can formulate the problem of traffic congestion as, given a record of traffic observations presented as sequence of tensors, predict observations some steps ahead into the future.

Using tensors allow further incorporation of CNN and LSTM for solving the forecasting of highways congestion. The data from spatially distributed sensors is treated as an image. The image features are learned using convolution operations and the series of images over time are trained using LSTM which means the replacement of matrix multiplications in the

input, forget and output gates of LSTM (8) with convolution operations denoted by * (9). Therefore, the state of a particular cell in the grid is determined by the input and past states of its neighbours' states.

$$\begin{aligned} \boldsymbol{i}^{(t)} &= \sigma(\boldsymbol{W}_i * \aleph^{(t)} + \boldsymbol{U}_i * H^{(t-1)} + \boldsymbol{V}_i \circ C^{(t-1)} + \boldsymbol{b}_i) \\ \boldsymbol{f}^{(t)} &= \sigma(\boldsymbol{W}_f * \aleph^{(t)} + \boldsymbol{U}_f * H^{(t-1)} + \boldsymbol{V}_f \circ C^{(t-1)} + \boldsymbol{b}_f) \\ \boldsymbol{o}^{(t)} &= \sigma(\boldsymbol{W}_o * \aleph^{(t)} + \boldsymbol{U}_o * H^{(t-1)} + \boldsymbol{V}_o \circ C^{(t)} + \boldsymbol{b}_o) \\ C^{(t)} &= \boldsymbol{f}^{(t)} \circ C^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tanh(\boldsymbol{W}_c * \aleph^{(t)} + \boldsymbol{U}_c * H^{(t-1)} + \boldsymbol{b}_c) \\ H^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh(C^{(t)}) \end{aligned} \tag{9}$$

where all inputs $\aleph^{(1)}, \aleph^{(2)}, ..., \aleph^{(t)}$, cell outputs $C^{(1)}, C^{(2)}, ..., C^{(t)}$, hidden states $H^{(1)}, H^{(2)}, ..., H^{(t)}$, and gates $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$ and $\boldsymbol{o}^{(t)}$ are 3-dimensional.

2.2. Performance Indices

In the experiment, the performance of the traffic flow prediction models are measured by four indicators MAE, RMSE, MRE and ACC presented by (10 – 13). MRE is one of the most common indicators for comparing prediction accuracy. However, when the traffic flow value is large, the MRE will be small. Therefore, MAE, RMSE and ACC were used to supplement the error analysis. Their expressions are presented in Eqs. 10-13:

$$MAE(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} | \hat{y}_i - y_i | \tag{10}$$

$$RMSE(\hat{y}, y) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{11}$$

$$MRE(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} \frac{| \hat{y}_i - y_i |}{y_i} \tag{12}$$

$$ACC(\hat{y}, y) = (1 - \frac{1}{N} \sum_{i=1}^{N} | \frac{\hat{y}_i - y_i}{y_i} |) \text{ x } 100\% \tag{13}$$

where N is the length of prediction, $\hat{y}$ and y_i represent the predicted and measured (observed) value for the i^{th} sample.

3. Experiments

Data source for this project is average speed collected by sensors installed on highways in California – a system known as the Performance Measurement System (PeMS) [25]. It is a professional traffic acquisition system with 15,000 detectors installed in California. The sensors are installed either in the pavement of the roadway or are radar- or video-based data collection systems. The sensors detect the presence of a vehicles recording relevant traffic data every 30 s and store them in the database. Two adjacent sensors placed at a fixed distance can detect the same vehicle twice and measure speed of the vehicle.

Each of these highways has several links. A link can be defined as a stretch of roadway between two adjacent interchanges/exits of a highway. Each link has speed averaged over a 5-minute time period for several days. The sections of highways in Los Angeles County, namely, I-5, US-101, CA-170, CA-134, CA-2, CA-110 were chosen for this study can be seen in Fig 1.

A sample plot for three sensors for about one day is shown in Fig. 2. It shows that the 'peak' period, represented by low speed, can occur during different times of the day. Once compiled, the data was scaled / normalized. Both the StandardScaler using mean and standard deviation and MinMax scalers were used. As the MinMaxScaler provided better training results, it was used for all further experiments and analyses.

Although deep learning-based models do not usually contain as a step data stationarity check, we included it after the scaling process for several sensor time series as performance in learning and model prediction are better for stationary time series. We perform the stationarity test using unit root Dickey-Fuller algorithm [24]. The *p*-value for all our tests was close

to 0, which implies that there is no unit root among any of the series. The details of Dickey-Fuller test for one sensor are: Test statistic -1.457574e+01; p-value 4.552138e-27; #Lags (length of sequences) used 2.200000e+01; Number of observations used 3.424900e+04.

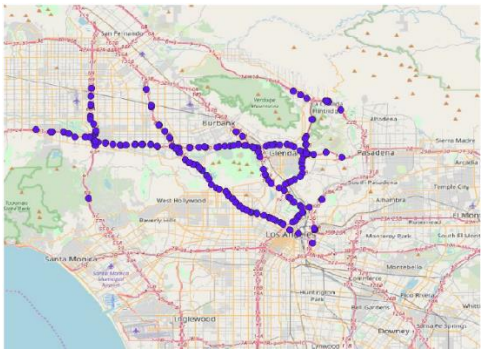


Fig. 1. Highway map with sensor locations

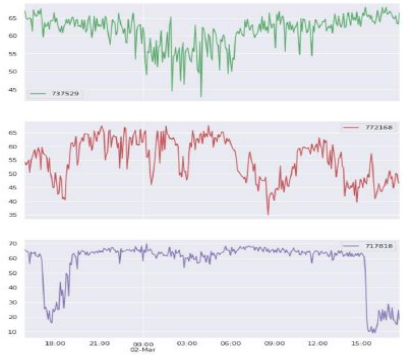


Fig. 2. Three sensors speed observations

3.1. Experimental Settings

Dataset. The data was extracted for all 207 sensors from the PeMS data warehouse for the duration between February and June 2017 taken at a 5-minute interval for months between February and June collected and compiled into a data table. 70% of the available time period of data is used for training, 20% for validation and 10% for testing. Although validation dataset is ignored in many NNs with DL research experiments, we consider its importance as an unbiased evaluation of the model that is a fit on the training set while tuning hyperparameters and allowing visualization of training performance. The validation subset is a crucial part of training the model to avoid overfitting and underfitting which improves the test results of the chosen topology. Besides this, validation accuracy can be used to determine when to perform certain actions during training. For example, callback function such as EarlyStopping or ModelCheckpoint in Keras can stop the training process when the model stabilizes, and no more epochs contribute to the model learning as well as to save the weights of the model at its peak.

3.2. Experimental Models and Results

Several deep learning neural network architectures were explored and evaluated for their prediction accuracy using Python, Keras and Tensor Flow. All experiments besides the proposed CSLTM include different DL topologies as GRU and STLM where parameters as number of layers, neurons, epochs and number of previous lags used to predict are tuned for best performance. CLSTM is used with different kernels and number of filters, stacked GRU with one, two and three hidden layers having number of neurons between 64 and 128 is implemented based on balance between accuracy and number of parameters. Also, models using different sequence lengths as input are tested as well as models with three past observations used to predict future observation. The predictions are made for 15, 30 and 60 minutes in future and each is evaluated across the different models. Additionally, the plots of predicted travel time are compared with observed travel time to demonstrate predictions accuracy. The performance is tested against the two baseline models: historical average and VAR. Historical average is calculated using the average of the speed observations for the same time period of the same day in the previous week. The VAR model for predicting the speed for all the N sensors s steps in the future denoted as ($\boldsymbol{X}^{t+s} \in R^N$) uses observations in the past L steps and is represented by Eq.14.

$$X^{t+L} = \beta_1 X^t + \beta_2 X^{t+1} + \ldots + \beta_S X^{t-L+1} + \varepsilon \tag{14}$$

For the first experimental set the number of epochs for training all participating DLNNs is set to 40, batch size to 128 and activation function to rectified linear unit (ReLU). Table 1 shows experimental results of different architectures, the number of parameters, the prediction loses with MAE and RMSE as well as details of baseline models.

Analysis of Table 1 shows that the proposed CLSTM models demonstrates the best prediction performance regardless of the number of layers, number of filters and the choice of testing error. Among the rest of the single layered architecture models, some LSTM models performed better when compared to GRU. However, GRU two and three stacked architectures and different choice of parameters demonstrate improvement when compared with LSTM.

Table 1. Performance evaluation of DLNNs and baseline models

Model - Architecture	# Parameters	Prediction Loss (MAE) 3 Step (15 min)	Prediction Loss (RMSE) 3 Step (15 min)
Historical average		**0.1**	**0.15**
VAR		**0.12**	**0.21**
GRU - 1 layer - 64	20,704	0.086	0.1598
GRU - 1 layer - 128	155,727	0.0792	0.1458
LSTM - 1 layer - 64	83,087	0.0811	0.1526
LSTM - 1 layer - 128	70,535	0.0808	0.1449
GRU - 2 layers - 64	90,447	0.0779	0.1584
GRU - 2 layers -128/64	179,535	0.0766	0.1537
LSTM - 2 layers -128/64	234,895	0.0782	0.1606
GRU - 2 layers - 256/128	530,895	0.0754	0.1487
GRU - 3 layers - 256/256/128	924,879	0.0743	0.1484
CLSTM - 1 layer / 20 filters / 2x2	13,767	0.0736	0.1184
CLSTM - 1 layer / 20 filters / 3x3	23,027	0.0731	0.1176
CLSTM - 1 layer / 40 filters / 3x3	74,847	0.0730	0.1176
CLSTM - 2 layers / 20 filters / 3x3	49,667	0.0726	0.1174

The sensor predictions for different models were compared with the observed data in Fig. 3(a) for the purpose of visualization. It can be seen that GRU 1 layer performs fairly well except that at one point in time the prediction of average speed is negative, which is physically impossible. The best model CLSTM with 2 layers is closest to the observed test data and is able to closely predict the peak period where the observed speed (in mph) starts to drop. Another instance comparing the predictions is shown in Figure 4(b). As in the previous case, CLSTM model is able to predict the inception of peak period better than other models. Also, in this case, based on the observed data, it appears that around time unit 220-225, the peak period is about to finish and the observed speed recovers to free flowing speed. However, at time unit 230 (denoted by an arrow), we observe another speed drop. Usually this happens when there is an accident. The CLSTM 2-layer model was able to correctly predict this situation.

The cases where CLSTM may not detect the second onset of congestion might be because the architecture is not learning congestion from other sensors. As the neighbouring sensors are close to each other, the congestion detected at one sensor could reach the adjacent sensor in longer time due to the nature of traffic flow. Thus, there is correlation between speeds observed at neighbouring sensors. This spatial correlation needs to be exploited and added to the temporal learning of NNs to obtain predictions with better accuracy.

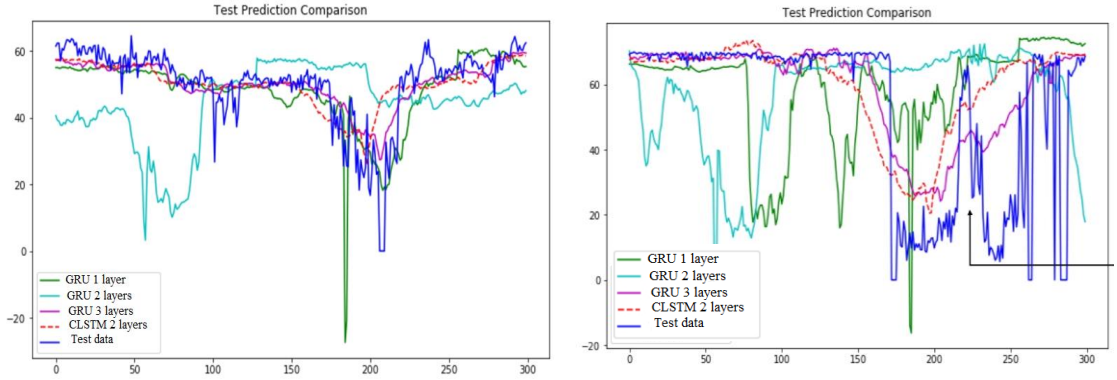
Due to the higher volume of computational operations related to CLSTM architectures, the next set of experiments which we demonstrate in this paper include a small subset of seven neighboring sensors on highway US-101 of the sensor system. The prediction performance related to data peak detection is shown in Fig. 4. Fig. 4(a) shows that the CLSTM with 2 layers accurately predicts the peak and test data well. In addition, CLSTM and stacked GRU are also able to predict few hours of missing data of the sensor as shown by the flat line between 80 and 150 time units in Fig. 4(b).

4. Conclusion

Traffic congestion problem is an important application for enabling traffic information dissemination and a key use case in 'smart cities'. The prediction of traffic congestion is performed over PeMS sensor data for highways in California using over four months of speed data collected every 5 minutes. Traffic congestion prediction is a time series prediction problem that can be mapped on to sequence prediction modeling framework in deep learning. Two different test metrics for comparing the DL architectures are applied.

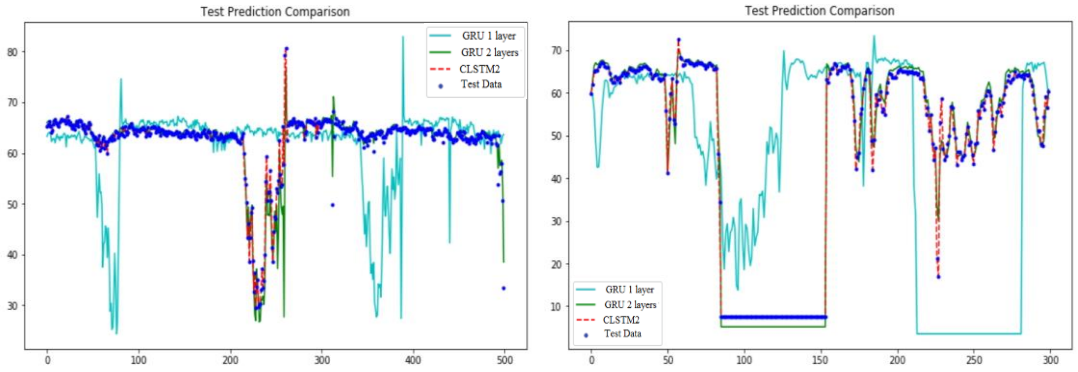
In this paper we implement the proposed CLSTM and several types of DL RNNs in solving traffic prediction. Their performance is evaluated and compared with two baseline models. Our experiments verify that CLSTM demonstrates the best predictions, while the stacked GRU (2-layer model with 128 and 64 neurons using 3-lags) performed better than LSTM.

At the same time in solving the formulated problem we observed that there are instances where particular DL model is not so close to observed data. The reason could be that the model may not have learned specific features on speed reduction and correlations that might be observed between other links. This can be particularly addressed by using the spatial proximity of sensors as an additional input. The impact of other factors as weather, incident etc. on traffic flow data needs to be further studied in order to improve the prediction accuracy.



(a) (b)

Fig. 3. Prediction (speed in mph) comparison across models



(a) (b)

Fig. 4. Comparison of prediction CLSTM vs GRU architectures

References

[1] D. Schrank, B. Eisele and T. Lomax, "2019 Urban Mobility Report", Texas A&M Transportation Institute, Texas, USA, August 2019.

[2] E. Cascetta, *Transportation System Engineering: Theory and Methods*, vol.49, Springer Science&Business Media, 2013.

[3] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and X. Xie, "Discovering spatiotemporal causal interactions in traffic data streams". In SIGKDD, 2011, pp. 1010–1018.
[4] I. Okutani, I.and Y.J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory" in Transp. Res. Part B Methodology, 1984, 18, pp. 1–11.
[5] Y. Xie, Y. Zhang and Z. Ye, "Short-Term Traffic Volume Forecasting Using Kalman Filter with Discrete Wavelet Decomposition", in Computer-Aided Civ. Infrastruct. Eng. 2007, 22, 326–334.
[6] R. X. Zhong, J. C. Luo, H. X. Cai, A. Sumalee, F. F. Yuan, and A.H.F. Chow, "Forecasting Journey Time Distribution with Consideration to Abnormal Traffic Conditions" Transportation Research Part C: Emerging Technologies, 2017. 85: 292-311
[7] Y.H Feng, J. Hourdos and G. A. Davis, "Probe Vehicle based Real-Time Traffic Monitoring on Urban Roadways", in Transportation Research Part C: Emerging Technol., 2014. 40: 160-178.
[8] W.W. Qin and M. P. Yun, "Estimation of Urban Link Travel Time Distribution Using Markov Chains and Bayesian Approaches" in Journal of Advanced Transportation, Article ID 5148085, 14 pages, 2018.
[9] Y. Bengio, I. Goodfellow and A. Courville, *Deep Learning*, MIT Press, 2016.
[10] W. Huang, G. Song, H. Hong, and K. Xie, "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning", IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 2191–2201, 2014.
[11] Lv, 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, vol.16, no.2, pp.865–873, 2015.
[12] X. Ma, H. Yu, Y. Wang and Y. Wang Y, "Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory", 2015, doi:10.1371/journal.pone.0119044] https://doi.org/10.1371/journal.pone.0119044.
[13] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", 2014, arXiv: 1406.1078.
[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory", in Neural Computing, 1997, Vol 9, pp 1735–1780.
[15] C. Olah, C, "Understanding LSTM Networks", 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/].
[16] X. Ma, 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, 2015, 54, pp.187–197.
[17] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction", in Proceedings of the Chinese Association of Automation, Hefei, China, 19–21, May 2017; pp. 324–328.
[18] J. Wang, F. Hu, L. Li, " Deep Bi-directional Long Short-Term Memory Model for Short-Term Traffic Flow Prediction:, in Proceedings of the Neural Information Processing 24th International Conference, Guangzhou, China, 14–18 November 2017; pp. 306–316.
[19] Y. Hou and P. Edara, " Network Scale Travel Time Prediction using Deep Learning", in Transportation Research Record 2018, Vol. 2672(45), 115-123.
[20] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting", 2015, arXiv: 1506.04214.
[21] Y. Li, R. Yu, C. Shahabi and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting", Presented at the ICLR, 2018, arXiv:1707.01926
[22] M. Bouchouia and F. Porter, "High dimensional regression for regenerative time-series: an application to road traffic modeling", October 2019, arXiv: 1910.11095v2.
[23] W. Wei, H. Wu and H. Ma, "An Autoencoders and LSTM-Based Traffic Flow Prediction Method", Sensors, 2019, 19, 2946; doi:10.3390/s19132946.
[24] D. A. Dickey and W. A. Fuller, W. A., "Distribution of the Estimators for Autoregressive Time Series with a Unit Root" in Journal of the American Statistical Association, 1979, 74 (366): 427–431.
[25] D. Rumelhart, G. Hinton and R. Williams, R. "Learning representations by back-propagating errors" in *Nature*, 1986, 323, 533–536.
[26] Caltrans, Performance Measurement System, 2019. Available online: http://pems.dot.ca.gov (accessed 11/2019)