Short-term Travel Time Prediction by Deep Learning: A Comparison of Different LSTM-DNN Models

Yangdong Liu, Yizhe Wang, Xiaoguang Yang *, Linan Zhang
The Key Laboratory of Road and Traffic Engineering, Ministry of Education, Tongji University
Shanghai 201804, China
E-mail: {liuyd, 16wangyizhe, yangxg}@tongji.edu.cn,
nathan_zhangln@qq.com

***Abstract*—Predicting short-term travel time with considerable accuracy and reliability is critically important for advanced traffic management and route planning in Intelligent Transportation Systems (ITS). Short-term travel time prediction uses real travel time values within a sliding time window to predict travel time one or several time step(s) in future. However, the nonstationary properties and abrupt changes of travel time series make challenges in obtaining accurate and reliable predictions. Recent achievements of deep learning approaches in classification and regression shed a light on innovations of time series prediction. This study establishes a series of long short-term memory neural networks with deep neural layers (LSTM-DNN) using 16 settings of hyperparameters and investigates their performance on a 90-day travel time dataset from Caltrans Performance Measurement System (PeMS). Then competitive LSTM-DNN models are tested along with linear models such as linear regression, Ridge and Lasso regression, ARIMA and DNN models under 10 sets of sliding windows and predicting horizons via the same dataset. The results demonstrate the advantage of LSTM-DNN models while showing different characteristics of these deep learning models with different settings of hyperparameters, providing insights for optimizing the structures.**

Keywords—prediction, travel time, deep learning, long short-term memory neural networks (LSTM), deep neural networks (DNN)

I. INTRODUCTION

The short-term prediction of traffic flow parameters including flows, speeds, densities and travel time compose an important major part in development and application of Intelligent Transportation Systems (ITS), forming the foundation of Advanced Traffic Management Systems (ATMS) and providing essential supports for travel guidance and route planning of travelers. A time series of traffic flows or travel time generally has similarity among its inner sub-series corresponding to the same weekdays (e.g., Monday in last week to Monday in next week), weekends and same quarters. Despite the periodicity and seasonality, however, these time series are always being affected by various external factors including traffic crashes, fluctuation in traffic demand and abrupt changes of weather conditions, which results in nonstationary properties and brings difficulties to the prediction.

The short-term travel time prediction refers to predicting travel time of a certain road section or a route for a short period of time, from several minutes to hours. Consider a time series of travel time $X = \{x_1, x_2 \cdots, x_T\}$, short-term prediction aims to predict $X_m = \{x_{T+1}, \cdots, x_{T+m}\}$ in next time step(s) $\{T+1, \cdots, T+m\}$ while $m \geq 1$. Here m, representing the number of time steps to predict, is generally addressed as forecasting horizon or prediction horizon. Prediction horizon can contain one time step or several time step of prediction. One-step prediction is mostly adapted in short-term time series prediction because the accuracy is relatively more possible to be guaranteed. In general, the more time steps for prediction, the more possible the accuracy to decrease. Full part or a subpart of travel-time series X is used to do prediction. Online or real-time prediction can then be achieved when a sliding time window of a certain length moves along the time axis and processes a subpart or a section of a time series at each time steps. The short-term travel time prediction in this paper is real-time.

To now, an amount of different kinds of appoaches have been studied and applied to short-term traffic flow parameters predicion. These methods can generally be seperated into two groups: 1) statictical methods, and 2) machine learning methods. In statistical methods, autoregressive integrated moving average (ARIMA) model is mostly common used for doing traffic flow parameters prediction. [1] and [5] used ARIMA models for freeway travel time prediction. Several variants of ARIMA models have been proposed since the first attempt of using ARIMA for freeway travel time prediction in [1]. Ref. [11] proposes H-ARIMA+ model to incorerate more patterns of data within peak hour and spatial influence from traffic events when considering the possible failure in predicting during peak hours and in effect of events as the traffic flow parameters values are only treated as generic time series. In [14], A spatial-temporal ARIMA (STARIMA) model is addressed for integrating spatial and temporal information to predict traffc flow. Ref. [6] presents an Bayesian network approach for traffic flow prediction. A Bayesian dynamic linear model is proposed for dynamic short-term travel time predicion in [8]. In [13], a local online kernel method for Ridge regression is developed for predicting urban travel times. For machine learning methods, the most popular are artificial neural networks (ANN) and support vector regression (SVR), a regression method based on support vector machine (SVM). A SVR based travel time prediction method is proposed in [3] and ANN based methods can be seen in [4] and [9]. Some approaches also incorporate statistical and machine learning methods and traffic flow theories to conduct modeling and predicting of travel time data. For instance, Ref. [12] utilize bottleneck identification algorithm from traffic flow theories and incorporate clustering, stochastic congestion maps, online congestion searching methods to conduct prediction of freeway travel time using both historical and real-time data.

On the other hand, deep learning methods as a branch of machine learning studies have recently drawn massive attentions from worldwide researchers separated in various field, since a big breakthrough for training deep neural networks (DNN) was proposed in [7]. These methods form a certain amount of state-of-the-art solutions to many classification and regression problems in several different subjects, in which recurrent neural networks (RNN) are now often being used for modeling sequential data, such as time series data and video or audio data. Long short-term memory (LSTM) units (cells) and gated recurrent units (GRU) are now widely adapted as alternatives of simple recurrent units to form a whole recurrent neural network and to model or learn complicated information contained within sequential data. A number of studies on time series and other sequential data that applied deep learning approaches indicated the advantage property of LSTM neural networks and other deep learning methods. In transportation research area, the early attempt of using deep learning methods is [15], which presents a deep belief networks (DBN) method to predict freeway traffic flow. Ref. [16] proposes a stacked autoencoder (SAE) model for freeway traffic flow prediction. In [17], LSTM networks are used for the first time for traffic flow parameters prediction and are successively be experimented in several studies such as [18]. [20] and [21] respectively proposes a DBN model and a DNN model for traffic speed and traffic flow prediction. Ref. 19 develops an error-feedback recurrent convolutional neural networks (eRCNN) structure, incorporates spatial and temporal relationship to conduct traffic speed prediction for ring roads in Beijing, China.

Despite the studies cited above using deep learning methods for traffic time series prediction, these studies did not go deeper into studying the optimization of structure design and hyperparameter settings of deep learning models in use for traffic data. It is important and essential because the designs of deep learning models are variable for learning different type of data. The design is more flexible when compared to traditionally well-used models and therefore it can also be more difficult to achieve an absolutely optimal effect of model establishment. Hence, this paper is aimed to explore how the "accurate" performance could change in quantity with varying settings of LSTM neural networks with deep fully connected layers. We call these models LSTM-DNN deep learning models. In the meantime, we compare these models with commonly used time series regression/prediction models as well.

Specifically speaking,

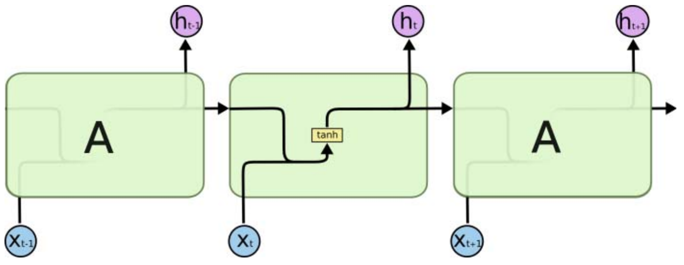
- we investigate short-term prediction results of travel time using 16 different LSTM-DNN models by arranging and combining 3 important hyperparameters including the number of LSTM cells and layers, and of deep fully connected layers;
- we select competitive LSTM-DNN models from above and test them along with linear models and machine learning models such as linear regression, Ridge and Lasso regression, ARIMA and DNN models in several experiment groups containing 10 sets of sliding windows and predicting horizons to furtherly check the accuracy of both deep learning and well-used prediction methods.

The organization of this paper is as follows. In Section II the structures of LSTM-DNN models for short-term travel time prediction are introduced, along with LSTM cells and deep fully connected layers. Section III presents 2 groups of experiments, with the first groups of experiments for comparing and selecting LSTM-DNN models, and second groups of experiments for testing LSTM-DNN models and benchmark methods in the combinations of 2 sliding windows and 5 prediction horizons. Finally, Section IV makes a conclusion and analyzes future work.

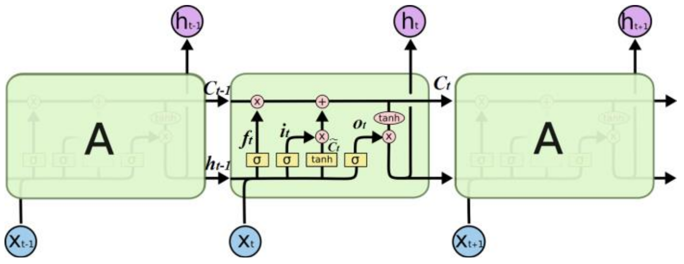
II. The LSTM-DNN Models

A. *Long Short-term Memory (LSTM) Cells*

Artificial neural networks (ANN) can generally be divided into two categories, i.e. feed forward neural networks (FFNN) and recurrent neural networks (RNN). a FFNN does not have recurrent or loop connections within its cells rather than RNN. And FFNN are applied earlier and wider in short-term time series prediction than the latter, but the results differ from different research. A FFNN that is usually called back propagation neural network (BPNN) contains 1 or 2 hidden layers. In order to obtain more accurate results from FFNN, the length of input data should be expanded and more layers should be added to itself. But FFNN is stopped from expanding the length of input layer or adding far more hidden layers due to the problem of gradient explosion and gradient vanishing until it was solved firstly by Hinton in [7]. Since then, a FFNN can be trained with wider length of input layers and deeper hidden layers (2 or more, from dozens to hundreds can be implemented), which forms deep neural networks (DNN). And now a DNN with fully connected layers can be trained to get more accurate results in short-term time series prediction. On the other hand, the architecture of RNN is naturally suitable for modeling sequence data because the recurrent connections within each neural cells enable the cells to map the input of sequential data and output from each of them. And RNN was also faced with the problem of gradient vanishing along with the RNN sequence. Then, long short-term memory (LSTM) cell was developed to solve the gradient vanishing problem [3], which then replaced simple recurrent cells and are widely used as a variant of simple recurrent cells in RNN structure for now.



(a) simple recurrent cells using tanh activation function



(b) LSTM cells using tanh activation function

Fig. 1. The architectures of simple recurrent cells and LSTM cells

The main improvement of LSTM cells against simple recurrent cells is the forget gates added. As Fig.1 shows, a simple recurrent cell only contains an activating function (here the default is $\tanh(\cdot)$) aside from input and output, while a LSTM cell has a forget gate and update gate more. The forget gate allows a LSTM cell to remove or add needed information to the cell state.

Let the input travel-time series in a sliding window be $X=\{x_1,x_2\cdots,x_T\}$, the hidden vector sequence $H=\{h_1,h_2\cdots,h_T\}$ is calculated then and an output sequence $Y=\{y_1,y_2\cdots,y_T\}$ is given by the LSTM network sequence. The following equations should be iterated:

$$h_t = H(W_h\cdot[h_{t-1},x_t]+b_h) \tag{1}$$

$$y_t = W_y h_t + b_y \tag{2}$$

where W is weight matrices (e.g. W_h is input-hidden weight matrix), b is bias vectors. $H(\cdot)$ is hidden layer function and is iterated by the following function:

$$f_t=\sigma\left(W_f\cdot[h_{t-1},x_t]+b_f\right) \tag{3}$$

$$i_t=\sigma\left(W_i\cdot[h_{t-1},x_t]+b_i\right) \tag{4}$$

$$\tilde{C}_t=\tanh\left(W_C\cdot[h_{t-1},x_t]+b_C\right) \tag{5}$$

$$C_t=f_t\cdot C_{t-1}+i_t\cdot\tilde{C}_t \tag{6}$$

$$o_t=\sigma\left(W_o\cdot[h_{t-1},x_t]+b_o\right) \tag{7}$$

$$h_t=o_t\cdot\tanh\left(C_t\right) \tag{8}$$

where $\sigma(\cdot)$ is sigmoid function, $\sigma(\cdot)$ and $\tanh(\cdot)$ are respectively define in Eq. (9) and Eq. (10). f,i,o,c are respectively forget gate, input gate, output gate and cell update gate.

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{9}$$

$$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}} \tag{10}$$

B. The LSTM-DNN models for prediction

Again, the historical travel-time series in current time step T is $X=\{x_1,x_2\cdots,x_T\}$, the problem is to predict travel time x_{T+m} while $m\geq1$, indicating either predict value in next time step or in several time steps ahead. And the input data $X_{input}=\{x_{T-(k-1)},\cdots x_T\}$ has total length of $k\left(1\leq k\prec T\right)$, which is the length of sliding window for prediction.

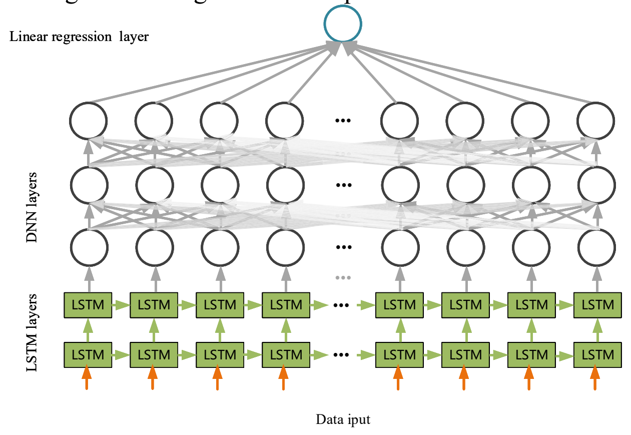


Fig. 2. The architectures of a LSTM-DNN model

As illustrated in Fig.2, the proposed LSTM-DNN models are composed of

1) one (or stacked) LSTM layers (shown in green blocks): the number of the layers can be 1 to many, 2 LSTM layers are stacked as an example in Fig.2;

2) deep fully connected neural layers (DNN layers, shown by gray circle units): the number of the layers can be 1 to many, 3 DNN layers are exemplified in Fig.2; and

3) a deep neural layer with 1 neural unit with linear activation (shown by blue circle unit). The output only has 1 value of travel time, so only 1 unit exits in this layer and this layer is not considered as the DNN layers part mentioned above.

Data are directly input to the first LSTM layer. The orange color arrows shown under the bottom LSTM layer only represent where to input the data. The outputs from a LSTM layer cannot be used directly as prediction results because in each time step LSTM yields output with the same length of input data (e.g. have a dimension of timesteps × 1). Hence, One or several DNN layers should be added above the LSTM layer for transferring output dimension. So how many DNN layers should be added to obtain best predictions becomes a problem to be study. The size of each stacked LSTM layer can be various and unique. Second, the size of each DNN layers should better be equal but can also be design with various values and unequal with each other.

The objective function is set as mean square error (MSE) as Eq. (11), where x_i denotes the actual travel time in training data and $\hat{x}_i$ is the prediction values. We use adaptive subgradient methods developed in [10] to train the proposed models. The entire network can be established by using Tensorflow, a recently developed deep learning framework in Python.

$$MSE = \frac{1}{n}\sum_{i=1.}^{n}\left(x_i - \hat{x}_i\right)^2 \tag{11}$$

III. Experiments

A. Dataset

The data for this study are obtained from Caltrans Performance Measurement System (PeMS), an online system that provides traffic data collected over 16,000 loop detectors located in California, the U.S. We collected travel time data on Corridor 13 (Alameda I-80) freeway section. The data duration is 90 days from Sep 18th 00:00, 2016 to Dec 17th 23:59 2016. Travel time values are aggregated per 5 mins, so there are 288 data points for one days and 25,920 points in total. The data were split into training (70%, 63 days), validating (10%, 9 days) in order to train the LSTM-DNN models and testing (20%, 18 days) data for investigate the prediction ability of the models since testing data are never seen by the models during training. As for using statistical methods, there is no need of validating data so the data were split into training (80%) and testing (20%) data for training statistical models such as linear regression, Ridge regression , Lasso regression and ARIMA.

B. Evaluation Metrics

Mean absolute percentage error (MAPE) and root mean square error (RMSE) are employed as evaluation metrics for our travel time prediction models. x_i denotes the actual travel time and $\hat{x}_i$ is the prediction values.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|x_i - \hat{x}_i\right|}{x_i} \tag{12}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1.}^{n}\left(x_i - \hat{x}_i\right)^2} \tag{13}$$

C. Hyperparameter Settings for LSTM-DNN Models

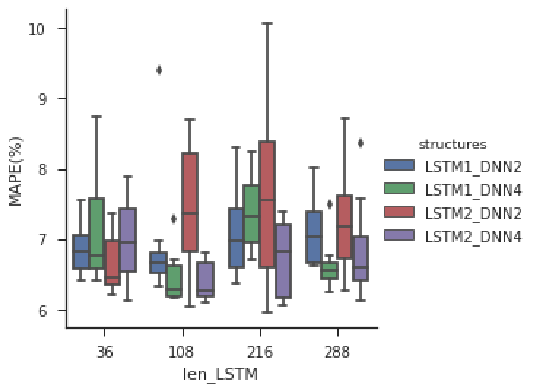
In this experiments we intended to create a highly challenging prediction mission for deep learning models. After several preliminary experiments, we set the input data length only be 6 timesteps (6 × 5 mins = 30 mins), and the task is to predict travel time at 12 timesteps ahead (60 mins ahead). From section 2 we can find that there are three important hyperparameters for LSTM-DNN models: 1) the number of LSTM units, addressed as "len_LSTM" (here for simplification, we make equal size of each LSTM layers for stacking), we set 4 values increasing with same interval; 2) number of LSTM layers (n_LSTM), we chose only two values – 1 and 2; and 3) number of DNN layers (n_DNN), two options – 2 and 4 – are provided. The summary of hyperparameter settings is in TABLE I.

In fact, various choices can be made towards the selection of these parameters and extremely a wide range of numbers can be used. But to leverage the training complexity and grained accuracy, we only used these values for experiments. Note that each number chosen for len_LSTM can exactly be associated with integer hours, e.g. 36 × 5 mins = 3 hours, and 216 × 5 mins = 18 hours. Besides, the learning rate, number of epoches, and batch size for each times training one single model are equal and are respectively set to be 0.001, 80 and 180 based on experience from several preliminary experiments.

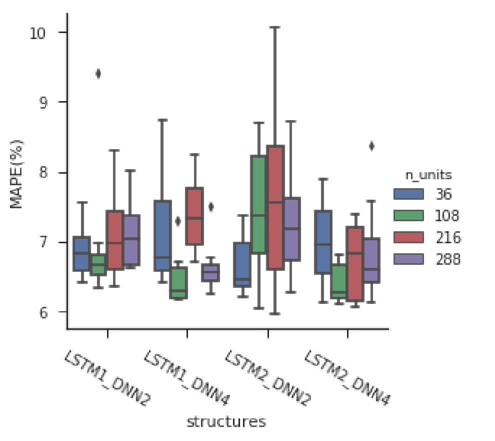
TABLE I. Hyperparameter Settings

Sliding window length	30 mins (6 time steps)
Prediction horizon	60 mins (12 time steps)
Number of LSTM units (len_LSTM)	[36, 108, 216, 288]
Number of LSTM layers (n_LSTM)	[1,2]
Number of DNN layers (n_DNN)	[2,4]

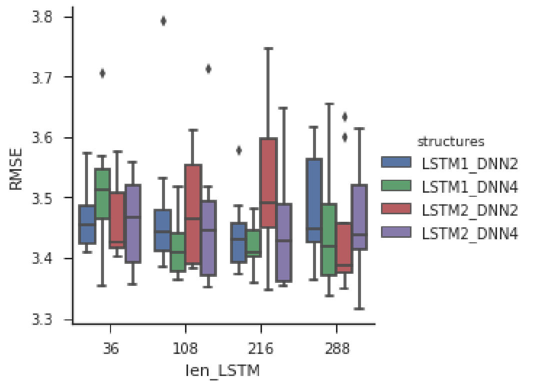
16 individual LSTM-DNN models with different combination of the 3 hyperparameters have been established and each model are trained 8 times separately, and tested 8 times separately



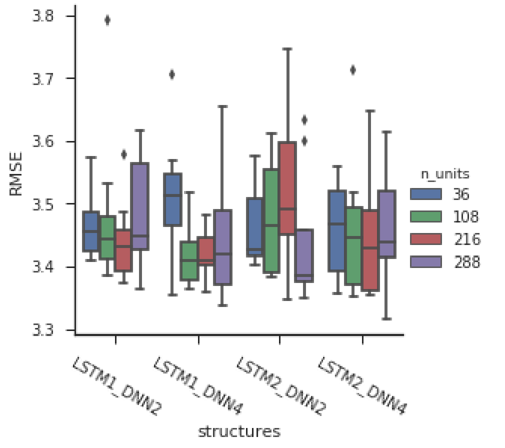
(a) MAPE(%) on test data (by len_LSTM)



(b) MAPE(%) on test data (by structures)



(c) RMSE on test data (by len_LSTM)



(d) RMSE on test data (by structures)

Fig. 3. Evaluation metrics of prediction on test data (8 groups of 16 experiments)

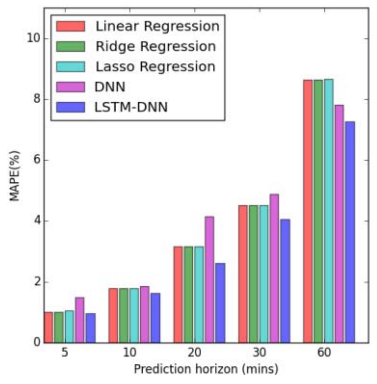
as well. So, 8 groups of 16 experiments (totally 128 times of experiments) have been conducted for comparing prediction results. Each experiment contains a training and a testing phase.

The training process for each group is kept the same. The reason for conducting multiple time experiments is that the parameters get from a trained deep learning model can be different each time, thus the prediction results vary.

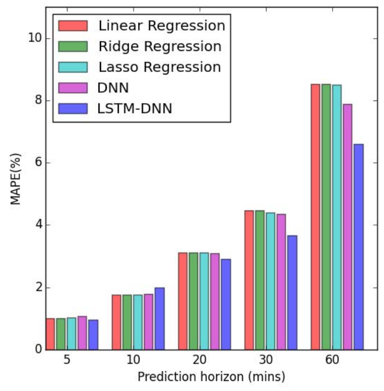
We illustrated the testing phases' MAPEs and RMSEs for 8 groups in Fig.3. The legend "LSTM1_DNN2" denotes that the number of LSTM layers(n_LSTM) equal to 1 and the number of DNN layers (n_DNN) 2, and so forth. For convenience we use style LSTM 1 + DNN 2 to describe the model structures.

Fig.3 indicates that better results cannot always be obtained when making deeper or wider neural networks. For MAPE, it is clear from Fig.3(a) that when len_LSTM is 36, combination LSTM 2 + DNN 2 is better than LSTM 2 + DNN 4, but when len_LSTM increases to 108, 216 and 288, performance from the latter becomes better as seen from the median levels. More interestingly, when len_LSTM is the biggest (288), LSTM 1 + DNN 4 is relatively better than LSTM 2 + DNN 2 (or + DNN 4) as the box of distribution is narrower. In overall scope, LSTM 1 + DNN 4 with 108 len_LSTM and LSTM 1 + DNN 4 with 288 len_LSTM get more centralized low MAPEs and hence are the relatively best two models with the measurement of MAPE. Also, from Fig.3(b), a phenomenon should also be paid attention that models with moderate len_LSTM (i.e., 108) are more stable and get more centralized low MAPEs than the others in 3 out of 4 structures. When the structure becomes LSTM 2 + DNN 2, all 4 len_LSTM types of models become dramatically unstable, indicating LSTM 2 + DNN 2 structure possibly an impropriate one for prediction tasks since it lacks robustness.

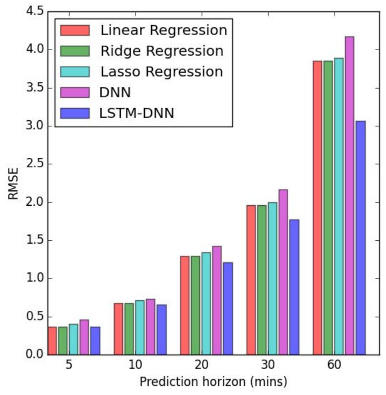
When comparing with RMSE in Fig.3(c) and Fig.3(d), models with 1 LSTM layers get better results (more centralized distribution and lower median) than those with 2 LSTM layers in 128 experiments. And the results from models with 36 and 108 len_LSTM are more stable, despite of outlier models. That exit in all 4 types of len_LSTM groups. Furthermore, LSTM 1 + DNN 4 with 108 len_LSTM gets the best performance and robustness in 8 groups. The smallest values of MAPEs and RMSEs are more likely to be obtained from models with 216 and 288 len_LSTM, but correspondently, the larger values have higher probabilities to be seen in these models. And no matter for MAPE or RMSE, when len_LSTM is 36, LSTM 1 + DNN 2 models are always better than LSTM 1 + DNN 4 according to the box widths and levels, but this situation gets contrary with the increasing of len_LSTM (or at least almost equal). Therefore, LSTM 1 + DNN 4 with 108 len_LSTM is overall-optimal choice among those models in both precision and robustness.



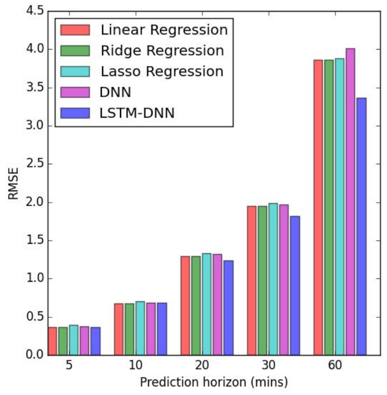
(a) MAPE(%) on test data (60 mins as sliding window)



(b) MAPE(%) on test data (30 mins as sliding window)



(c) RMSE on test data (60 mins as sliding window)



(d) RMSE on test data (30 mins as sliding window)

Fig. 4. Evaluation metrics of prediction on test data

D. Comparision between LSTM-DNN Models and Benchmarks

In order to compare the prediction ability of LSTM-DNN with well-used statistical and machine learning models, we continually conducted two groups of totally 10 experiments, in which sliding window length is set to be 12 and 6 (i.e. 60 mins and 30 mins), and prediction horizon is [1,2,4,6,12] time steps (i.e. [5,10,20,30,60] mins), such that 10 experiments can be conducted with each sliding windows length paired to one prediction horizon. In each experiment 6 types of model are trained and tested, which are respectively 3 types of linear models, an ARIMA model, a DNN model (4 fully connected layers with 288 units in each) and a LSTM-DNN model. The utilized linear models are 1) linear regression, this is the simplest model in our experiments, 2) Ridge regression and 3) Lasso (least absolute shrinkage and selection operator) regression. The linear models and ARIMA model are trained on training data (80%) and predict on test data (20%). The test data remain unknown for these models until the data are used for testing the models. The LSTM-DNN model we chose here is LSTM 1 +DNN 4 with 108 len_LSTM, which is analyzed to be the best model in all LSTM-DNN alternatives in part C. The α for Ridge regression and Lasso regression is respectively set to 0.5 and 0.05, as this setting can leverage the output of MAPE and RMSE and yield better prediction results on test data in preliminary experiments. All additional hyperparameters of the LSTM-DNN model remain same with which in part C,. LSTM-DNN model is trained 6 times for each of the 10 experiments to get the best result and DNN models also trained several times for each prediction scenario.

From the results shown in Fig.4 and TABLE II, we can clear see the advantages and drawbacks of our LSTM-DNN model. Both prediction performance in short and long horizon the deep learning model almost gained every best result among the benchmarks models. When the sliding window is relatively narrowing down and prediction horizon stretching, the advantage of the proposed model is dramatically being revealed. For MAPE or RMSE, only one non-optimal case of LSTM-DNN model occurs when using travel time within 60-min sliding windows to predict 10-min travel time in future. When focusing on ARIMA model, one important condition that need to be clarified is that the experiments used iterative prediction on test data. Rather than directly doing multiple timestep prediction like other models, ARIMA model can only do 1-step prediction directly, if it is required to do multi-step prediction ARIMA model generally takes iterative method, i.e. use the prediction value as feedback to predict the following timestep values. To increase the prediction quality, iterative prediction often retrains the trained ARIMA model when a new true value is fed back to it, such that the ARIMA model updates its parameters in every timestep. However, for maintaining consistency of the experiments it is not allowed to use test data to retrain the models. Thus, a ARIMA model conduct i-step prediction only using its own (i-1) step prediction values, which leads to the gradually increasing errors and the huge deviation of prediction in long horizon prediction. So the results of ARIMA are not plotted on Fig.4.

When focusing on DNN models, the MAPEs are smaller than the other benchmarks when the horizon is long enough (cases like 60-min for 60-min, and 30-min for >20 min prediction) but DNN models have no advantages in terms of RMSEs. The possible reason is that the causal relation of the timestep dimension is partially ignored by DNN units when inputting data into them.

TABLE II. EVALUATION METRICS OF PREDICTION ON TEST DATA

	MAPE(%), Sliding window = 60 mins					**MAPE(%)**, Sliding window = 30 mins				
Prediction horizon(mins)	5	10	20	30	60	5	10	20	30	60
Linear Regression	1.010187	1.781746	3.157644	4.513085	8.637864	1.006194	1.773854	3.122103	4.457734	8.528927
Ridge Regression	1.010173	1.781706	3.157573	4.513007	8.637823	1.006172	**1.773804**	3.122005	4.457624	8.528827
Lasso Regression	1.048166	1.793461	3.165147	4.504833	8.671028	1.028043	1.769604	3.107643	4.407132	8.498129
DNN	1.493290	1.861104	4.151781	4.876978	7.820807	1.065725	1.778982	3.084992	4.358483	7.877626
ARIMA	1.208255	2.484198	5.562116	9.407818	23.736445	1.130722	2.236957	4.790908	7.990758	20.229357
LSTM-DNN	**0.967767**	**1.621363**	**2.616827**	**4.057407**	**7.264099**	**0.961468**	1.997938	**2.920018**	**3.655459**	**6.600965**
	RMSE, Sliding window = 60 mins					**RMSE**, Sliding window = 30 mins				
Prediction horizon(mins)	5	10	20	30	60	5	10	20	30	60
Linear Regression	0.367289	0.677546	1.298362	1.954813	3.853114	0.366843	**0.677064**	1.296032	1.952448	3.858874
Ridge Regression	0.367294	0.677550	1.298375	1.954831	3.853134	0.366848	0.677065	1.296040	1.952459	3.858887
Lasso Regression	0.404177	0.711253	1.340855	1.996254	3.887231	0.397264	0.703925	1.329985	1.984265	3.884927
DNN	0.459849	0.731222	1.427575	2.161596	4.173776	0.377937	0.687101	1.319625	1.968736	4.011477
ARIMA	0.402316	0.792926	1.687567	2.735245	4.004502	0.388855	0.750618	1.554574	2.497897	5.806814
LSTM-DNN	**0.362902**	**0.655964**	**1.211798**	**1.771794**	**3.064785**	**0.365456**	0.688558	**1.234101**	**1.822048**	**3.364138**

IV. CONCLUSION AND FUTURE WORK

In this paper we firstly investigate short-term travel time prediction effects of 16 LSTM-DNN models by arranging 16 groups of combination for three important hyperparameters including the number of LSTM cells and layers, and of deep fully connected layers (DNN layers) and conducted 128 experiments. We find that adding depth and width is not always the best choice to obtain better results. Maintaining a middle scale of deep learning model that appropriate to certain data, in turn, is more possible for generating expected accurate and solid prediction results according to the evaluation metrics of MAPE and RMSE. Secondly, we choose competitive LSTM-DNN models from above and test them along with linear prediction models such as linear regression, Ridge and Lasso regression, ARIMA and DNN models in several experiment groups containing 10 combinations of sliding windows and predicting horizons to furtherly check the accuracy of both deep learning and well-used statistical and machine learning methods. The results indicate the strong power of our LSTM-DNN models when the sliding window is narrow and the prediction horizon becomes far ahead, both MAPEs and RMSEs are lower than benchmark methods under these scenarios.

The results from the comparison of LSTM-DNN models also reveal the complexity and difficulty in the optimization for deep learning prediction models. We mainly focus on three important hyperparameters related to the structure of LSTM-DNN models but several more factors also have impacts on model structure optimization for producing optimal prediction results, such as optimizing methods selection, and the setting of learning rate, epoch amount, batch size and optional regularization for training a model. We mainly employed a set of appropriate default values for these hyperparameters in order to simplify the comparison, emphasize the key aspects for building models and reduce time cost. It should be a huge work for optimization when considering changes of all these hyperparameters, which is still an unsolved problem in deep learning area. Nevertheless, the further work should be focusing on more available structures of deep learning model, increasing experiments, adding more well-used models for comparison and optimizing the structure of deep learning models under different prediction scenarios. Besides, specific prediction effects should be evaluated aside from the evaluation metrics. That is, to look deeper into the shapes of predicted time series to check how well a model can accommodate abrupt changes to be predicting.

References

[1] M. S. Ahmed and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*, 1979, no. 722. ARIMA

[2] S. Hochreiter and J. Schmidhuber , "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997.

[3] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, pp. 276–281, 2004. SVR

[4] J. V. Lint, S. Hoogendoorn, and H. V. Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, pp. 347–369, 2005. ANN

[5] D. Billings and J.-S. Yang, "Application of the ARIMA Models to Urban Roadway Travel Time Prediction - A Case Study," *IEEE Int. Conf. on Systems, Man and Cybernetics*, 2006, pp. 2529–2534. ARIMA

[6] W. Zheng, D.-H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *J. Transportation Eng.*, vol. 132, pp. 114–121, 2006. Bayesian network

[7] G. E. Hinton, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, pp. 504–507, 2006.

[8] X. Fei, C.-C. Lu, and K. Liu, "A Bayesian dynamic linear model approach for real-time short-term freeway travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 19, pp. 1306–1318, 2011.

[9] R. Li and G. Rose, "Incorporating uncertainty into short-term travel time predictions," *Transportation Research Part C: Emerging Technologies*, vol. 19, pp. 1006–1018, 2011. ANN

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[11] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing Real-World Transportation Data for Accurate Traffic Prediction," *IEEE 12th Int. Conf. on Data Mining*, 2012, pp. 595- 604. H+ARIMA

[12] M. Yildirimoglu and N. Geroliminis, "Experienced travel time prediction for congested freeways," *Transportation Research Part B: Methodological*, vol. 53, pp. 45–63, 2013. bottleneck identification algorithm, clustering, stochastic congestion maps, online congestion searching

[13] J. Haworth, 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, pp. 151–178, 2014.

[14] P. Duan, G. Mao, C. Zhang, and S. Wang, "STARIMA-based traffic prediction with time-varying lags," *IEEE 19th Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1610–1615. STARIMA

[15] W. Huang, G. Song, H. Hong, and K. Xie, "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, pp. 2191–2201, Oct. 2014.

[16] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, pp. 865–873, Apr. 2015.

[17] 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.*, vol. 54, pp. 187–197, May 2015.

[18] Y. Tian and L. Pan, "Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network," *IEEE Int. Conf. on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 153–158.

[19] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method,". *IEEE 16th Int. Conf. on Data Mining (ICDM)*, 2016, pp. 499–508.

[20] Y. Jia, J. Wu, and Y. Du, "Traffic speed prediction using deep learning method," *IEEE 19th Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1217–1222.

[21] H. Yi, H. Jung, and S. Bae, "Deep Neural Networks for traffic flow prediction," *IEEE Int. Conf. on Big Data and Smart Computing (BigComp)*, 2017, pp. 328–331.