**LSTM**

**Description**

Deep learning is a subset of machine learning based on artificial neural networks. Among them, recurrent neural network (RNN) and Long short-term memory (LSTM) like networks structures are widely used in time series analysis. LSTM is based on RNN architecture, and it can process single data points and entire sequences of aggregate data. RNN cannot solve the problem with long-term temporal dependencies, while LSTM units include a set of gates to control the information for long periods. Therefore, LSTM represented a deep learning methodology is selected to predict urban travel times.

Figure 1 shows the structures of LSTM. Each LSTM cell receives input from input sequence (data set), previous cell state and output. In a LSTM cell, there are three gates: Forget gate, input gate and output gate. Forget gate is responsible for what information should be forgotten. Input gate is to control how much information should be fed from the input sequence. Output gate computes output from cell state to be sent to the next cell (i.e. Short term memory). Apart from that, the cell state from previous LSTM cell also pass to next cell with editing some information from the current cell. In this case, current LSTM cell can still obtain information from long time period cells (i.e. Long term memory).

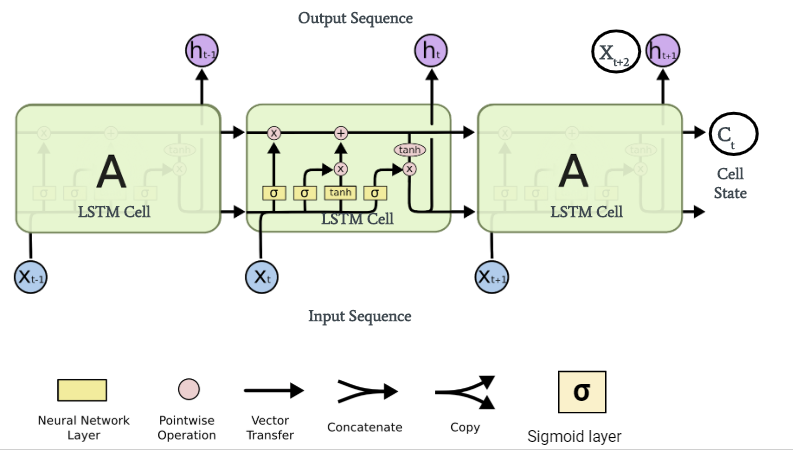


Figure 1. LSTM flow chart

The following equations are updating formulas for each gate:

Forget gate:

Input gate:

Output gate:

Long term memory:

Short term memory:

Where, *W* and *b* are the weight matrices and bias vectors of that gates and need to be learned during the training process. During the training process, the data will be fed into the LSTM model, and calculate the *W* and *b* of each cell. The iteration will be finished until *W* and *b* become stable.

**Parameter tuning**

Link 1745 is selected to tune parameters and other links in this study inherit the same values from link 1745. The input dataset of LSTM requires three dimensions dataset contains . The feature in this study is only urban travel times and equal to one. From PACF of link 1745 fig (from Chloe part), the current time has a significant relationship with the last six time steps. In other words, the current time result may relate to the previous six time steps at most. Therefore, the time step of LSTM model is six, and the first six urban travel values are the input of the next value without any transformations. The input data format is , and is the number of inputs.

There are two essential parameters of LSTM: number of units and learning rate. Parameter of number of units is the size of hidden layer in a LSTM cell (i.e. the number of neurons of each gate). The larger size of units will lead to high accuracy and longer computation time, while smaller size will cause underfitting. Demuth et al. (2014) proposed a formula to calculate the number of hidden neurons in neural network:

Where, = number of input neurons, = number of output neurons, number of samples in training data set and is an arbitrary scaling factor usually 2-10.

In this study, every time six data records are fed into predictor and = =6. The size of total original dataset is 30 days and 5400 records, and the first 23 days are training dataset. Therefore, the number of training data set is 4140. After calculation, the possible number of hidden neurons fall into the range of 34 to 172, and the possible values are [20,32,50,64,80,100,128] to tune LSTM model in this study.

Learning rate is responsible for the learning process of the model. Higher learning rate may have lower computation and unstable learning process, and a lower learning rate will lead to overfitting. Learning rate usually is 0.01 s for standard multi-layer neural network, and it should be linear or uniform sampling in the log-domain (Bengio, 2012). Therefore, it is assumed that the possible learning in this study is [0.001,0.003,0.005,0.008,0.01,0.03,0.05].

Grid searching is employed to find approximate values of these two parameters. The score of each parameter is mean square error (MSE) of the last iteration in development set. The results are in a heatmap () to visualize the result (Figure 2), and higher value indicates lower MSE and better performance. It can be seen that the result become worse when the number of units is less than 20, and higher learning rate will result in a better performance. In this study, the lowest MSE occur when the number of units is 128 and learning rate is 0.001. However, higher parameter value always leads to better performance, and lower learning may cause unstable results. The result at units 64 and learning rate 0.003 performance comparably high performance, and these values are selected as parameter values to save computation time and obtain stable results.

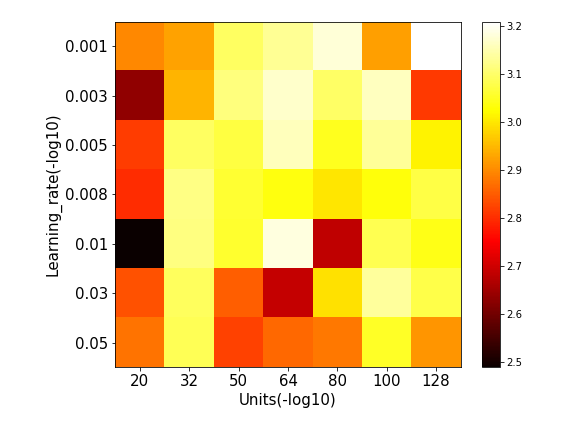


Figure 2. Heatmap results of LSTM parameter tuning

**Results**

The LSTM network in this study is a LSTM layer and one fully connected layer. 22 links make up the dataset and each link will have a model, there will be 22 LSTM models therefore. For training process, the number of epochs and batch size is required. These two parameters are from deep learning requirement instead of LSTM. Epochs are the number of propagations into the neural network, and one epoch indicates that the whole dataset has been passed once. A good performance model should have enough epochs to show a stable result and redundant number of epochs will cause unnecessary calculations. Batch size is the number of training examples utilized in one iteration. There are three modes of batch size choosing: batch mode (batch size is equal to the total dataset), mini-batch mode (between batch mode and stochastic mode) and stochastic mode (batch size is one). The lower batch size needs higher temporal requirement (calculation time), higher batch size leads to higher spatial requirement (memory).

The number of epochs will not have impacts on the LSTM performance, and it is assumed the epoch is equal to 35 in this study. The batch size may differ from the state of computers, and usually it is equal to the power function of 2 to easy calculated by GPU. In this study, the batch size sets to 128. It is assumed that 10% of training data is the development set, and the remaining is training set. Figure 3 shows the average MSE of 22 links against epochs, the MSE of training set and dev set is convergent, and the bias is lower with more epochs. Therefore, these models have a good bias-variance trade-off without overfitting, and the average MSE of 22 links dev set is 0.002.

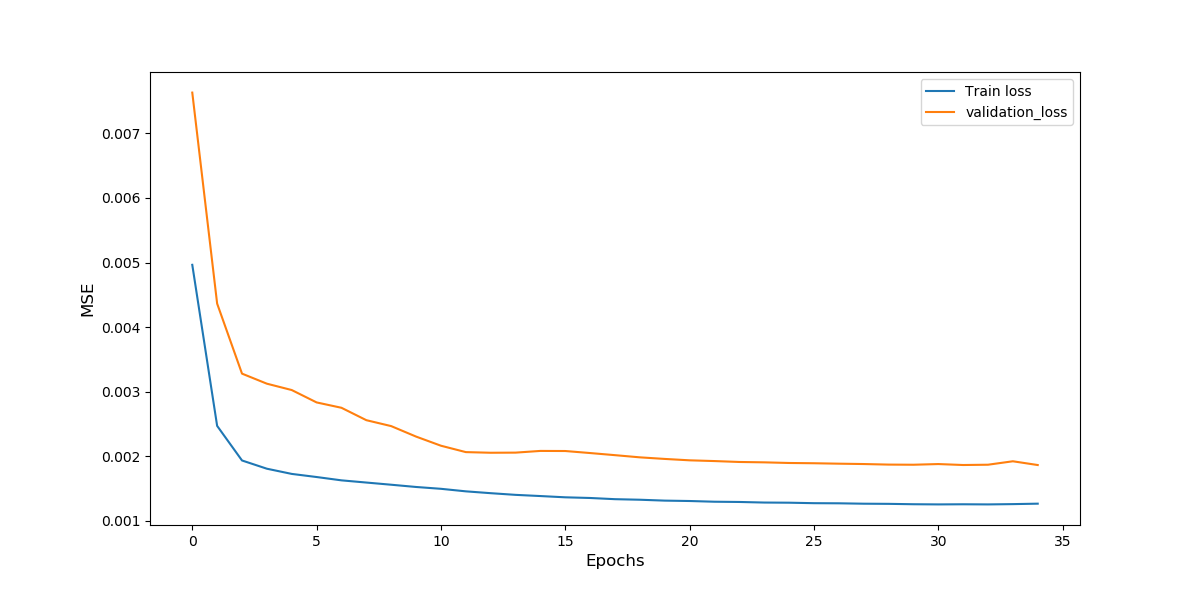


Figure 3. learning curve of LSTM

The data of last seven days is the testing set. There are 22 LSTM models, and the result of Link 1745 (The first model) on 24 January 2011 is shown in Figure 4 as an example. The predicted values show a similar tendency with true values, and both predicted and true values reach a peak at 15:00 and decline after 16:00. Figure 5 depicts the heatmap image representation of the true traffic speed and predictions by LSTM. It can be seen that the prediction result has a similar range of values with true values. The prediction result is smoother than the true values, which implies that there is still a difference between prediction result and true values.

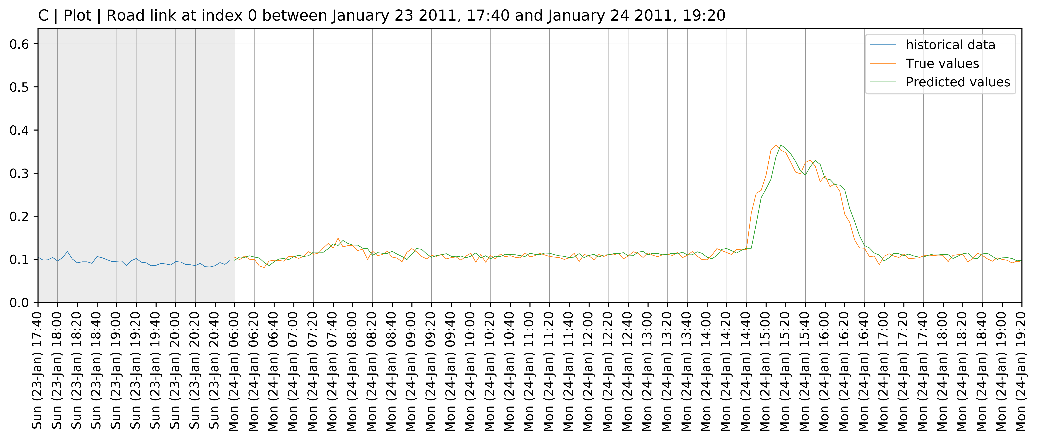


Figure 4. Predicting result of Link 1745 on 24 January 2011

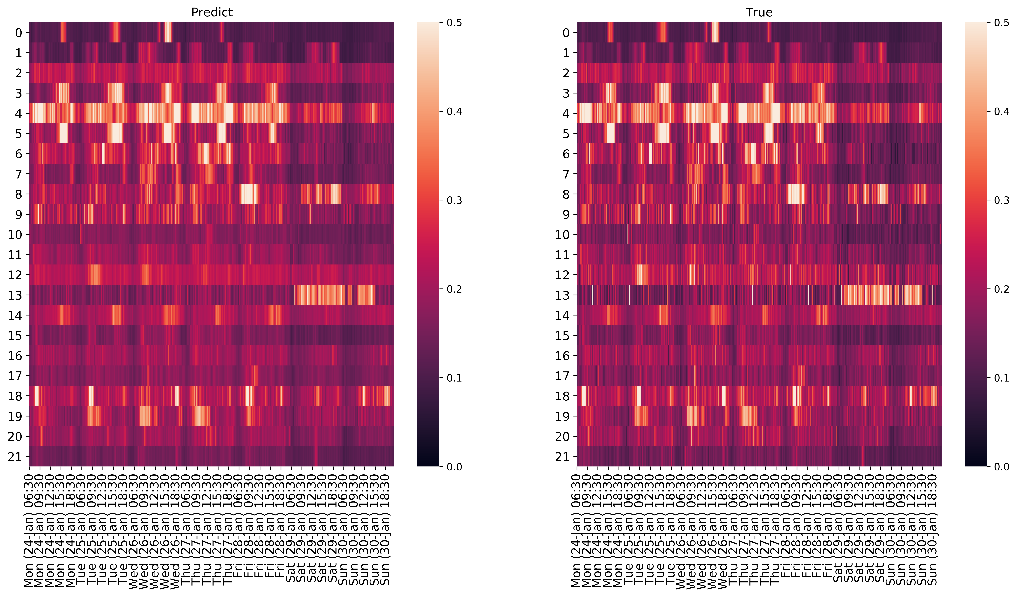


Figure 5. Predicting result in heatmap

To quantify the difference between prediction and true values, an evaluation metric is required. The data range of each model is different, and the evaluation result will be determined by higher data range models if average MSE is the performance metric. Therefore, MSE can only evaluate the performance in one model or in the training iteration process. In this case, average (coefficient of determination) is employed to evaluate the overall performance of 22 models. The following is defined equations of MSE and :

Where, , is predicted values and is the corresponding true values.

The final average of MSE of 22 LSTM models is 0.00158, and average of LSTM models is 0.348, which implies that 34.8% data can be explained by the model’s inputs therefore.

BENGIO, Y. 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures. *In:* MONTAVON, G., ORR, G. B. & MüLLER, K.-R. (eds.) *Neural Networks: Tricks of the Trade: Second Edition.* Berlin, Heidelberg: Springer Berlin Heidelberg.

DEMUTH, H. B., BEALE, M. H., DE JESS, O. & HAGAN, M. T. 2014. *Neural network design*, Martin Hagan.