Prediction of Station Level Demand in a Bike Sharing System using Recurrent Neural Networks

Po-Chuan Chen, He-Yen Hsieh, Xanno Kharis Sigalingging, Yan-Ru Chen, Jenq-Shiou Leu, *Senior Member, IEEE*
Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan
{M10202101, M10502103, M10307805, M10302154, jsleu}@mail.ntust.edu.tw

***Abstract*—Bike sharing systems have been widely applied to many cities and brought convenience to local citizens for short-ranged transportation. The bike shortage problem due to uneven bikes distribution is one of the biggest challenges in bike sharing systems. In this paper, we focus on station level prediction for each bike station. The proposed architecture is based on Recurrent Neural Network (RNN) and we use only one model to predict both rental and return demand for every station at once which is efficient for online balancing strategies. Without considering the global level bike distribution, the MAPE/RMSLE of the sum over the demand of each station may be too high for rebalancing strategies but the MAE/RMSE are satisficing at station level. Our evaluation shows that the proposed methods meet satisfied results at station level and global level in New York Citi Bike dataset.**

Keywords—Bike Sharing System; Demand Prediction; Station Level Prediction; Machine Learning; Deep Learning; Recurrent Neural Networks; Bi-directional RNN; Soft Attention Mechanism

I. Introduction

Bike sharing systems are getting popular nowadays. As a transportation system, a bike sharing system hosts many bike stations around the city, providing great convenience for citizens to rent bikes from nearest bike station and return them to other stations. There are some challenges in bike sharing systems. One of the challenges is the demand of each station can't be the same, which leads to unbalanced bikes distribution. For instance, someone rents bike from a station near his house and returns it to a station in front of his working place, but there are no dock available for returns at the latter station. When going home, the rental demand is higher than return demand at those stations where there are no bikes to rent. A heuristic solution is to move the bikes to rebalance the bike supply, for example by using trucks. However, the loading capacity of each truck is limited. To improve the efficiency of dispatch, the problem can be treated as finding the optimized routes for trucks as the rebalance strategy. Even though there are some researches on the prediction problems, most researches are about predicting global rental demand or rental demand at cluster level, not at station level which helps bike rebalance strategies most. Therefore, we propose the architecture based on recurrent neural networks to predict the future demand of the bike sharing system at station level. The proposed architecture uses only one model to predict bike rental and return demand. We evaluate our models in the New York Citi Bike dataset [1] and these models outperform baselines at station level and greatly improve at global level. Besides, our proposed architecture make prediction for all stations at once, which is highly efficiency for online process.

II. Recurrent Neural Networks

Most signals are time-series signals. In feedforward neural networks, each input is independent of each other and doesn't consider the input order dependency. Recurrent neural networks (RNNs) are designed to take input order into account and make it easy to handle time-series data. Figure. 1 shows a simple RNN and Eq. (1) and (2) as its equations.

$$h_t = f_h\,(W_h\,h_{t-1} + W_i\,x_t + b_h) \qquad (1)$$

$$\hat{y_t} = f_o\,(W_o\,h_t + b_o) \qquad (2)$$

W_h : the weight of hidden unit
h_{t-1} : hidden unit at time t-1
W_i : the weight of input unit
x_t : the input unit at time t
b_h : the bias of hidden unit
f_h : the activation function of hidden unit
h_t : hidden unit at time t
W_o : the weight of output unit
b_o : the bias of output unit
$\hat{y}_t$: the prediction at time t

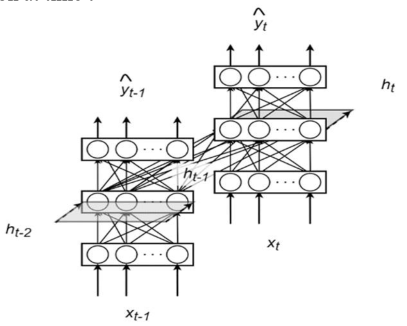


Figure 1. A Simple Recurrent Neural Network (RNN)

A. Loss Function

A loss function minimizes the errors between the output of a model and the target. It's a metric of error in output to tune the weights of our model and make the model more accurate. Different loss functions fit different problems. Generally, the formula of loss function can be expressed as Eq. (3) and used for regression. We discuss loss function deeply in section IV.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \qquad (3)$$

N : number of predictions
$\hat{y}_i$: prediction
y_i : ground truth

B. *Backpropagation Through Time*

The loss function gives us the error metric in output to help us understand the accuracy of our model, but it cannot improve the result. Backpropagation through time (BPTT) [2] helps us find out each weight being greater or less and minimize the loss function in an efficient way instead of randomly guess. In BPTT, we calculate the gradient of a loss function and propagate it with all the weights in backward direction. Therefore, we can tune all the weights in the right direction. We use BPTT to train RNNs as shown in Eq. (4). Basically, BPTT is not different from normal backpropagation except for taking the derivative of output loss of each timestamp with respect to weights.

$$\frac{\partial \mathrm{E}}{\partial \mathrm{W}} = \sum_{t=0}^{S} \frac{E_t}{W} \tag{4}$$

C. *Vanishing Gradient Problem*

The main problem in simple RNNs is that they suffer from vanishing gradient problem [3] when there are too many timestamps. In fact, the feedforward neural networks also suffer from vanishing gradient problem when they become deep. With specific initialization methods [4], the problem can be solved in feedforward neural networks, but it can't in RNNs. Fortunately, we have different types of RNNs that can solve this problem.

D. *Long Short-Term Memory*

Long short-term memory (LSTM) [5] in Figure 2 is a RNN architecture designed to address the vanishing gradient problem. Each LSTM unit has its own memory cell that can store information and use gates to control information flows. This gate control design helps the gradient flow through units to avoid from vanishing gradient.

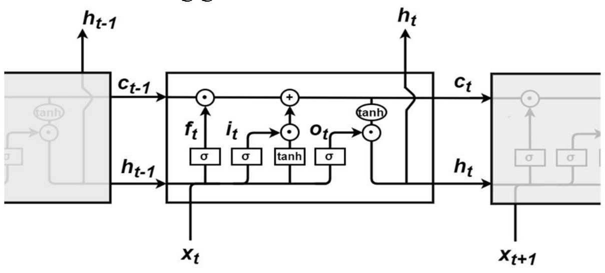


Figure 2. Long Short-Term Memory

First, we treat the input gate i_t and the forget gate f_t as the control gates of cell state c_t. With sigmoid function σ, the output value of input gate and forget gate are between 0 and 1. The input gate i_t decides how much input information add in, and the forget gate f_t controls how much information of previous cell state c_{t-1} flow through. The output gate o_t is the same here except for controlling the output of hidden state h_t.

During backpropagation, the forget gate f_t controls the gradient flow for both c_{t-1} and h_{t-1}. Compared with simple RNN, this gate control design eases the vanishing gradient a lot.

III. DATA EXPLORATION

Nowadays, bike sharing systems are very popular around the world. Many of them have released their datasets to public, mainly to be used for research purposes. We first considered to use the datasets from Taiwan owned Taipei YouBike [9]. However, the current release of YouBike datasets are quite limited. They only published the current status of bike stations instead of historical data. Therefore, we use the dataset from New York Citi Bike [1]. Citi Bike has released the historical data since its initial launch in 2013, and the dataset include trip level information revealing much information about the bike usage patterns. In this section, we explore the dataset in 2014, which has 8,081,216 individual trips.

TABLE I. ATTRIBUTES IN NEW YORK CITI BIKE DATASET

Trip Duration (seconds)
Start Time and Date
Stop Time and Date
Start Station Name
End Station Name
Station ID
Station Lat/Long
Bike ID
User Type (Customer = 24-hour pass or 7-day pass
user; Subscriber = Annual Member)
Gender (Zero=unknown; 1=male; 2=female)
Year of Birth

A. *Data Attributes and Basic Analysis*

The data attributes in Citi Bike dataset (Table I) provide us to analyze the trip patterns deeply. First, we analyze the available bike stations for each month to observe the system expanding speed.

The bike rental demand shown in Figure 3 starts increasing after February, and reaches highest point during summer and then decreasing. 90% of trips are contributed by subscribers and the other 10% are by customers shown in Figure 4 and 5. Comparing with the monthly average temperature, we find either the trips number or the percentage of trips by customers showing highly positive correlation with temperatures in Figure 6 and 7.

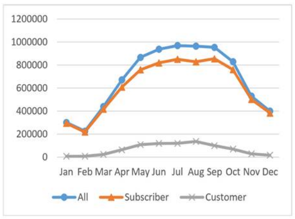


Figure 3. Number of trips(monthly)

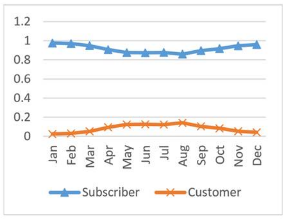


Figure 4. Percentage of trips by users (monthly)

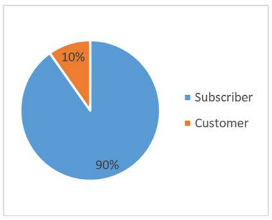


Figure 5. Percentage of trips by users (monthly average)

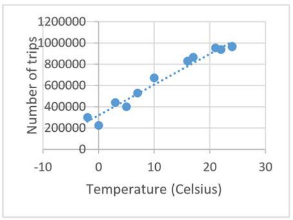


Figure 6. Number of trips vs. temperature (monthly)

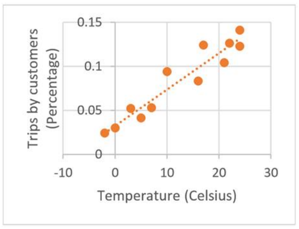


Figure 7. Percentage of trips by customers vs. temperature (monthly)

B. *Relationship between Weather and Bike Demand*

With temperature data shown in Figure 8, we find the temperature doesn't affect the rental demand during midnight but greatly influence it during the day.

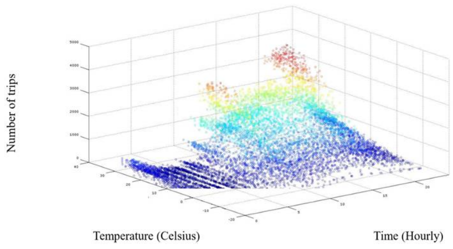


Figure 8. Number of trips vs. temperature vs. time (hourly)

The wind speed and humidity also show the correlation with rental demand shown in Figure 9. Unlike temperature, the wind speed and humidity have negative correlation with rental demand shown in Figure 10 and 11.

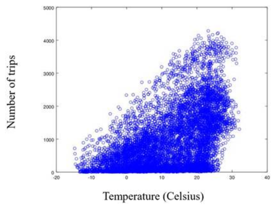


Figure 9. Number of trips vs. temperature (hourly)

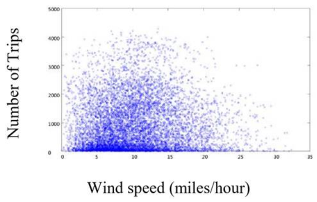


Figure 10. Number of trips vs. wind speed (hourly)

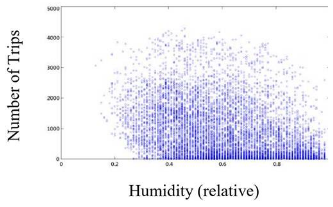


Figure 11. Number of trips vs. humidity (hourly)

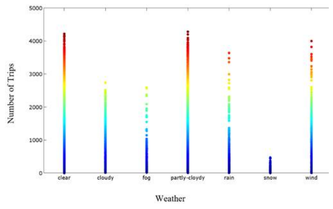


Figure 12. Number of trips vs. weather (hourly)

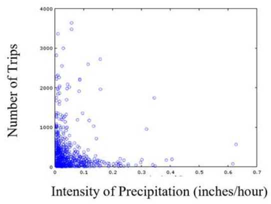


Figure 13. Number of trips vs. intensity of precipitation (rain, hourly)

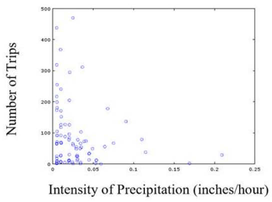


Figure 14. Number of trips vs. intensity of precipitation (snow, hourly)

Our weather dataset divides the hourly weather condition into seven categories: clear, cloudy, fog, partly-cloudy, rain, snow, and wind. With scatter plot in Figure 12, we find the rental demand of cloudy hour is around half of clear hour.

We can find the correlation between rental demand and precipitation in Figure 13 and 14 is negative and high rental demand during raining hour are outliers.

IV. Proposed Methods

The architecture proposed in this paper is based on recurrent neural networks (RNNs). This architecture can predict the bike rental demand for each of the station in the system. RNNs are sensitive to the time series data and capable of finding out the hidden temporal patterns. We know bike rental demand changes through the time and highly related with weathers, which is suitable for RNNs. RNNs work well on handling high-dimensional inputs and millions of weights detecting the hidden patterns, such as the inter-correlation among stations, and handle high-dimensional output so that the model can predict the number of check-in and check-out for all stations immediately.

A. *Data Preprocessing*

Table II shows the input data divided into three categories – time, weather and station. Feeding time information to the model helps it to learn historical patterns. Table III shows the prediction at station level. We transform the trip data from New York Citi Bike dataset into the number of check-in and check-out for each station during given time Δt (i.e. 5 minutes) and its corresponding value of next given Δt (i.e. 60 minutes) as prediction label.

TABLE II. Input Data Format

Category	Attribute	Value Type
Time	year	one-hot encode
	month	one-hot encode
	weekday	one-hot encode
	hour	one-hot encode
	quarter-hour	one-hot encode
Weather	hourly weather status	one-hot encode
	hourly temperature	real value
	hourly apparent temperature	real value
	hourly humidity	real value
	hourly intensity of precipitation	real value
	hourly probability of precipitation	real value
	hourly dew point	real value
	hourly wind speed	real value
	hourly visibility	real value
Station i (from 1 to N)	The number of check-in bikes from subscribers during Δt at station i	real value
	The number of check-out bikes from subscribers during Δt at station i	real value
	The number of check-in bikes from customers during Δt at station i	real value
	The number of check-out bikes from customers during Δt at station i	real value

TABLE III. OUTPUT DATA FORMAT

Category	Attribute	Value Type
Station i (from 1 to N)	The number of check-in bikes from subscribers during Δt at station i	real value
	The number of check-out bikes from subscribers during Δt at station i	real value
	The number of check-in bikes from customers during Δt at station i	real value
	The number of check-out bikes from customers during Δt at station i	real value

B. Loss Function

Since we want to predict both rental and return demand of each station, the multi-objective optimization problem can be written as a loss function in Eq. (5).

$$loss = \sum_{i=1}^{N}\left(\left|\hat{y}_{in,i} - y_{in,i}\right| + \left|\hat{y}_{out,i} - y_{out,i}\right|\right) + \alpha\left[log\frac{\left(\sum_{i=1}^{N}\hat{y}_{in,i}\right)+1}{\left(\sum_{i=1}^{N}y_{in,i}\right)+1} + log\frac{\left(\sum_{i=1}^{N}\hat{y}_{out,i}\right)+1}{\left(\sum_{i=1}^{N}y_{out,i}\right)+1}\right] \quad (5)$$

α : the parameter of global level MSLE
N: the number of stations
$\hat{y}_{in,i}$: the predict number of check-in bikes for station i
$\hat{y}_{out,i}$: the predict number of check-out bikes for station i
$y_{in,i}$: the ground truth number of check-in bikes for station i
$y_{out,i}$: the ground truth number of check-out bikes for station i

The first part of the equation is a station level MAE and the other part is a global level MSLE. The global level part works as the penalty to make the model learn the distribution of bike demand across all stations. We choose MAE instead of MSE because MAE is more stable to outliers and so does the case in MSLE vs. MAPE.

C. Proposed Architecture

The proposed architecture is shown in Figure 15. We use RNN as time-series encoder to capture the hidden temporal patterns. Because the output scale of either LSTM or GRU is between -1.0 and 1.0, the feedforward network on top of it can prevent the output values from being restricted. It is important to consider the size of training datasets to prevent overfitting. We use ReLu activation of the output layer instead of linear activation because all the predicted values are more than or equal to 0.

The shared-weight feedforward networks work as the input layer for each timestamp, which means each input networks are all identical. We can treat each identical input network as a bunch of weak classifiers to detect the same patterns for each timestamp and find out some useful local patterns before feeding into RNN.

The soft attention [6] is more like a soft selection that the network learns how to weight each RNN timestamps and transfer the weighted average to the output feedforward network.

We use the bi-directional RNN [7]. One direction is for normal forward states and the other direction is backward states. Consisting of these two, the network has the ability to scan the timestamps like grammatical analysis of sentences.

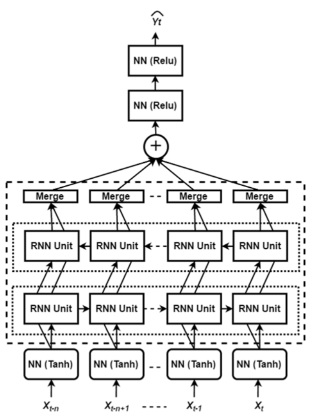


Figure 15. Shared Feedforward Neural Network + Bi-directional Recurrent Neural Network + Soft Attention Mechanism (SFNN+ Bi-RNN + SAM)

V. EVALUATION

We process the dataset of New York Citi Bike 2014 with 8,081,216 trip records and split it into a training set and a testing set. The training set contains the data from Jan. 1, to Sept. 30, and we use data from Oct. 1 to Dec. 31 as the testing set to evaluate our models. The data is then transform into the input format with Δt = 5 minutes and its corresponding number of check-in and check-out during next hour as ground truth label for prediction.

A. Settings of Proposed Architecture

Since each of the RNN unit stands for a given timestamp Δt = 5 minutes and 95% of trips are less than 50 minutes, the number of RNN units based on either LSTM or GRU is set to 12 for the proposed architecture, and the parameter α in the proposed loss function is fixed to 1.0 since the value of global level penalty is much lower than the value of station level. Due to time constraints, the number of training epochs is set to 50 for the proposed architecture.

B. Baseline Approaches

We choose three methods that all support multiple output regression as our baselines and also test on our proposed architecture with commonly used mean absolute error (MAE) loss function instead of our proposed loss function for comparison. For methods which cannot deal with time-series input data, we train those models with a given timestamp Δt = 5 minutes as a separate input instead of series of timestamps and train the models with the same given timestamp Δt = 5 minutes, but the input values are the sum over the last hour. The three methods are listed below.

1. Ordinary Least-Squares Regression (OLS).
2. Random Forest [8] with 50 estimators (RF)
3. Feedforward Neural Network, 4 layers with ReLu activations. (FNN)

C. Evaluation Metrics

For detailed comparison, we choose four metrics for evaluation. We use mean absolute error (MAE) in Eq. (6) and root mean squared error (RMSE) in Eq. (7) for station level evaluation, mean absolute percentage error (MAPE) in Eq. (8) for global level evaluation and RMSLE in Eq. (9) for both station and global level evaluation.

$$MAE = \frac{1}{M*N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\hat{y}_{ij} - y_{ij}\right| \tag{6}$$

$$RMSE = \sqrt{\frac{1}{M*N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\hat{y}_{ij} - y_{ij}\right)^2} \tag{7}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{8}$$

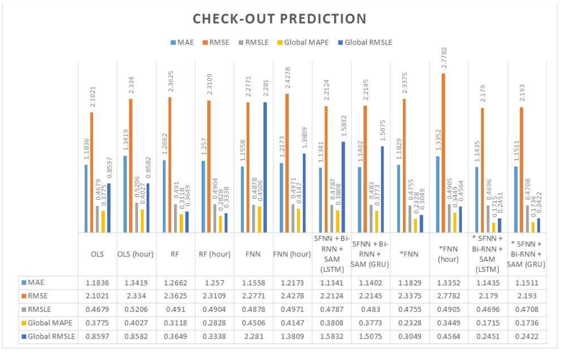
$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(log(\hat{y}_i + 1) - \log(y_i + 1))^2} \tag{9}$$

M : the number of stations
N : the number of testing samples
$\hat{y_i}$: prediction
y_i : ground truth

VI. Results

Figure 16 and 17 show the results of check-out and check-in prediction respectively. When we compare these two figures, all the baselines and the methods without the proposed loss function work well at station level, but the errors become huge at global level. Our methods outperform the baselines OLS (hour), RF, RF (hour), FNN and FNN (hour) at station level and outperform all the baselines greatly at global level. This shows our methods can perform well at station level and global level.

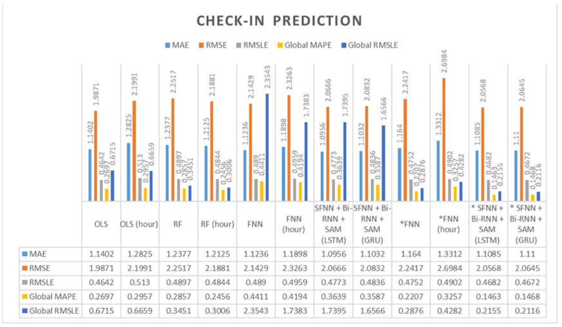
In general, the proposed architecture based on GRU outperform those based on LSTM.



	OLS	OLS (hour)	RF	RF (hour)	FNN	FNN (hour)	SFNN + Bi-RNN + SAM (LSTM)	SFNN + Bi-RNN + SAM (GRU)	*FNN	*FNN (hour)	* SFNN + Bi-RNN + SAM (LSTM)	* SFNN + Bi-RNN + SAM (GRU)
MAE	1.1836	1.3419	1.2662	1.257	1.1558	1.2173	1.1341	1.1402	1.1829	1.3352	1.1435	1.1511
RMSE	2.1021	2.334	2.3625	2.3109	2.2771	2.4278	2.2124	2.2145	2.3375	2.7782	2.179	2.193
RMSLE	0.4679	0.5206	0.491	0.4904	0.4878	0.4971	0.4787	0.483	0.4755	0.4905	0.4696	0.4708
Global MAPE	0.3775	0.4027	0.3118	0.2828	0.4506	0.4147	0.3808	0.3773	0.2328	0.3449	0.1715	0.1736
Global RMSLE	0.8597	0.8582	0.3649	0.3338	2.281	1.3809	1.5832	1.5075	0.3049	0.4564	0.2451	0.2422

Note: * indicates an architecture with the proposed loss function

Figure 16. Check-out prediction



	OLS	OLS (hour)	RF	RF (hour)	FNN	FNN (hour)	SFNN + Bi-RNN + SAM (LSTM)	SFNN + Bi-RNN + SAM (GRU)	*FNN	*FNN (hour)	* SFNN + Bi-RNN + SAM (LSTM)	* SFNN + Bi-RNN + SAM (GRU)
MAE	1.1402	1.2825	1.2377	1.2125	1.1236	1.1898	1.0956	1.1032	1.164	1.3312	1.1085	1.11
RMSE	1.9871	2.1991	2.2517	2.1881	2.1429	2.3263	2.0666	2.0832	2.2417	2.6984	2.0568	2.0645
RMSLE	0.4642	0.513	0.4897	0.4844	0.489	0.4959	0.4773	0.4836	0.4752	0.4902	0.4682	0.4672
Global MAPE	0.2697	0.2957	0.2857	0.2456	0.4411	0.4194	0.3639	0.3587	0.2207	0.3257	0.1463	0.1468
Global RMSLE	0.6715	0.6659	0.3451	0.3006	2.3543	1.7383	1.7395	1.6566	0.2876	0.4282	0.2155	0.2116

Note: * indicates an architecture with the proposed loss function

Figure 17. Check-in prediction

VII. Conclusion

In this thesis, we propose the architecture based on recurrent neural networks to predict bike demand for each station in a bike sharing system. Our proposed methods use a model which takes high dimensional time-series data from each station and uses hourly weather information as input to predict the next hour check-in and check-out demand for all station immediately.

Moreover, our results show that the MAPE/RMSLE of the sum over the demand of each station are so high if the methods do not take the global level demand into account even if the results of MAE/RMSE are satisfied at station level. When taking both station and global level demand into consideration, the propose methods meet satisfied results at station level and global level. In future work, we will focus on improving the current architecture by taking the advantage of trip information, such as trip duration, user type and start station, to predict the hourly bike demand at station to station level.

References

[1] Citi Bike. ‘New York City Bike Sharing System Data.’ URL: https://www.citibikenyc.com/system-data. Accessed: 21 April 2016

[2] Paul J. Werbos. "Backpropagation through time: what it does and how to do it." Proceedings of the IEEE 78.10 (1990): 1550-1560.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.

[4] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Aistats. Vol. 9. 2010.

[5] Sepp Hochreiter, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[6] Alex Graves. "Generating sequences with recurrent neural networks." arXiv preprint arXiv:1308.0850 (2013).

[7] Mike Schuster, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE Transactions on Signal Processing 45.11 (1997): 2673-2681.

[8] Leo Breiman. "Random forests." Machine learning 45.1 (2001): 5-32

[9] YouBike. ‘Taipei City Bike Sharing System Data.’ URL: http://data.taipei/opendata/datalist/datasetMeta?oid=8ef1626a-892a-4218-8344-f7ac46e1aa48. Accessed: 19 April 2016