The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Impact of Augmenting GRU Networks with Iterative and Direct Strategies for Traffic Speed Forecasting

Armando Fandango, R. Paul Wiegand, Liqiang Ni, Samiul Hasan
University of Central Florida
Orlando FL, USA
armando@ucf.edu

Abstract

In this paper, we report experimental results from augmenting Recurrent Neural Networks (RNN) with multi-step-ahead strategies for traffic speed prediction. For multi-step-ahead time series forecasting, researchers have applied MIMO, recursive, and direct strategies to machine learning methods in other domains. We applied the recursive and direct strategies to the GRU networks for predicting multi-step-ahead traffic speed and compared the prediction errors with the GRU network without these strategies (i.e. MIMO strategy). Based on the results from the experiments, we found that the direct strategy and the MIMO strategy produce models with smaller error metrics as compared to the recursive strategy. The direct strategy is computationally very expensive, thus MIMO strategy, i.e. the GRU models without any strategy, is our preferred recommendation.

Introduction

We have observed the emergence of deep learning based methods, specifically Recurrent Neural Network (RNN) architectures, for multi-step-ahead traffic flow predictions (Li, Wu, and Yoshinaga 2019; Tian et al. 2018; Yang et al. 2019; Luo et al. 2019; Rahman 2019; Rahman and Hasan 2018; Fu, Zhang, and Li 2016; Shao and Soong 2016; Chen et al. 2016; Tian and Pan 2015; Dai et al. 2017; 2019). While such research has generally indicated that adopting RNN architecture improves predictions for short-term traffic states, the impact of combining iterative and direct strategies with the RNN models has not been studied until recently (Fandango and Kapoor 2018; Fandango and Wiegand 2018).

In many different domains, the researchers have combined the direct and iterative strategies with classical machine learning models and derived varied conclusions, sometimes contradictory ones (Koesdwiady, Khatib, and Karray 2018; Koesdwiady and Karray 2018; Taieb and Hyndman 2012; Hamzaçebi, Akay, and Kutay 2009; Wen, Torkkola, and Narayanaswamy 2017; An and Anh 2015; Bontempi et al. 2013; Taieb et al. 2012; Taieb 2014; Tibbitt et al. 2013). For example, while comparing direct and recursive strategies, Chang et al. concluded recursive strategies produced better forecasts (Chang, Chiang, and Chang 2007). On the other hand, Hamzacebi et al. concluded that direct strategies produced better forecasts (Hamzaçebi, Akay, and Kutay 2009). These confusing conclusions and the lack of availability of research augmenting the RNN architectures with multi-step-ahead strategies motivates a strong desire to explore this area further.

In this paper, we present our investigation of the following question: *Does augmenting the RNN architectures with iterative or direct strategies for multi-step-ahead traffic speed prediction reduce error metrics such as MSE, MAPE and SMAPE?*. This paper extends recent work that used a smaller dataset (Fandango and Kapoor 2018; Fandango and Wiegand 2018) by adding a much larger data set from 88 detectors and by adding additional statistical analysis of the results.

Methods

Here we present a general description of the problem of multi-step-ahead traffic prediction, as well as the recursive, direct and MIMO strategies. In addition, we describe the recurrent neural network architecture we consider, gated recurrent unit architectures.

Multi-step-ahead Traffic Prediction

Traffic state prediction can be formally written as follows:

$$\{x_{t'}; t' = N+1, ..., N+h\} = f(\{x_t; t = 1, ..., N\}),$$

where:

- x_t is the observed characteristic of traffic state such as speed or volume at time t in a specific location in the transportation network
- $\{x_t; t = 1, ..., N\}$ is the given sequence of observations
- $\{x_{t'}; t' = N+1, ..., N+h\}$ is the predicted sequence for h number of future time steps; for multi-step-ahead $h \geq 2$, for one-step-ahead $h = 1$

Multi-step-ahead Strategies

In a *recursive strategy*, a single one-step-ahead model is trained on a fixed window of time steps (Algorithm 1), formally written as follows:

$$x_{t+1} = f_\theta\left(\{x_{t-w+1}, ..., x_t\}\,,\, \theta\right),$$

Algorithm 1: train with recursive strategy

```
Input  : $\{x_{t=1}, ..., x_{t=N}\}$
Output: $f_\theta$
1 Prepare data in rows of (features, label) pairs such
    that features = $\{x_{t-w+1}, ..., x_t\}$, label = $\{x_{t+1}\}$
2 for all (features, label) pairs do
3     find optimal $\theta$ such that $label \leftarrow f_\theta(features)$
```

where θ is the vector of model parameters and w is the window size or lag, i.e. number of past time steps to use as features.

The trained model f_θ is used to predict the value x_{t+1}, and the predicted value is appended to the input window for predicting the value at next time step. This is repeated until the value x_{t+h} is predicted (Algorithm 2).

Algorithm 2: predict with recursive strategy

```
Input  : $f_\theta, X = \{x_{t-w+1}, ..., x_t\}, h$
Output: $\{x_{t+1}, ..., x_{t+h}\}$
1 for $\delta \in \{1..h\}$ do
2     $x_{t+\delta} \leftarrow f_\theta(\{x_{t+\delta-w}, ..., x_{t+\delta-1}\})$
3     append $x_{t+\delta}$ to $X$
```

Here, if $w < h$ then the future forecasts start having only predicted values as input, thus accelerating the accumulation of prediction errors.

In the *direct strategy*, h number of one-step-ahead models are trained on fixed window of time steps to forecast δ^{th}-step-ahead (Algorithm 3), written as follows:

$$x_{t+\delta} = f_{\delta,\theta_\delta}\left(\{x_{t-w+1}, ..., x_t\}\,,\ \theta_\delta\right),$$

where θ_δ is the parameter vector of model $\#\delta$.

Algorithm 3: train with direct strategy

```
Input  : $\{x_{t=1}, ..., x_{t=N}\}, h$
Output: $\{f_{\delta,\theta_\delta}; \delta \in \{1..h\}\}$
1 for $\delta \in \{1..h\}$ do
2     Prepare data in rows of (features, label) pairs such
        that features = $\{x_{t-w+1}, ..., x_t\}$, label = $\{x_{t+\delta}\}$
3     for all (features, label) pairs do
4         find optimal $\theta$ such that
            $label \leftarrow f_{\delta,\theta_\delta}(features)$
```

Each of the δ^{th} trained model is used to predict the value δ^{th}-step-ahead (Algorithm 4), and the predictions are appended together to return the forecast vector.

The *multiple-input-multiple-output (MIMO) strategy* is the default strategy used if neither of the above strategies are employed. In this strategy, a single multi-step-ahead model is trained on a fixed window of time steps (Algorithm 5), formally written as follows:

$$\{x_{t+1}, ..., x_{t+h}\} = f_{h,\theta}\left(\{x_{t-w+1}, ..., x_t\}\,,\ \theta_h\right),$$

Algorithm 4: predict with direct strategy

```
Input  : $\{f_{\delta,\theta_\delta}; \delta \in 1..h\}, X = \{x_{t-w+1}, ..., x_t\}, h$
Output: $\{x_{t+1}, ..., x_{t+h}\}$
1 for $\delta \in \{1..h\}$ do
2     $x_{t+\delta} \leftarrow f_{\delta,\theta_\delta}(X)$
3 concat all $x_{t+\delta}$ to get the output
```

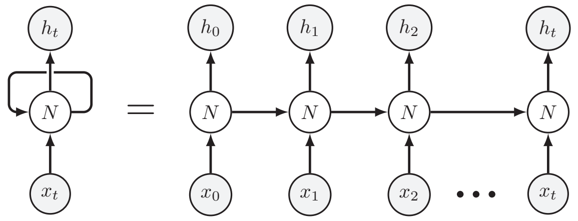

Figure 1: Recurrent Neural Network (RNN) Architecture

where θ_h is the parameter vector of the model.

Algorithm 5: train with MIMO strategy

```
Input  : $\{x_{t=1}, ..., x_{t=N}\}$
Output: $f_{h,\theta}$
1 Prepare data in rows of (features, labels) pairs such
    that features = $\{x_{t-w+1}, ..., x_t\}$, labels =
    $\{x_{t+1}, ..., x_{t+h}\}$
2 for all (features, labels) pairs do
3     find optimal $\theta$ such that
        $labels \leftarrow f_{h,\theta}(features)$
```

The trained model $f_{h,\theta}$ is used to predict the values $\{x_{t+1}, ..., x_{t+h}\}$ in one shot (Algorithm 6).

Algorithm 6: predict with MIMO strategy

```
Input  : $f_{h,\theta}, X = \{x_{t-w+1}, ..., x_t\}$
Output: $\{x_{t+1}, ..., x_{t+h}\}$
1 $\{x_{t+1}, ..., x_{t+h}\} \leftarrow f_{h,\theta}(X)$
```

Gated Recurrent Unit (GRU) RNNs

Recurrent Neural Network (RNN) architectures (Elman 1990) build on top of feed-forward neural networks by providing a mechanism for using the output for current state as input to next state within the same layer as shown in Fig. 1. This kind of architecture is well-suited to time series data because the inputs at next time step often depend on the previous time steps.

Deep RNN architectures suffer from the problem of vanishing and exploding gradients when more cells or hidden layers are added. The gradients gradually diminish to zero, or they become so large that they tend to approach ∞ (infinity). Hence, many variants of RNN have been proposed in

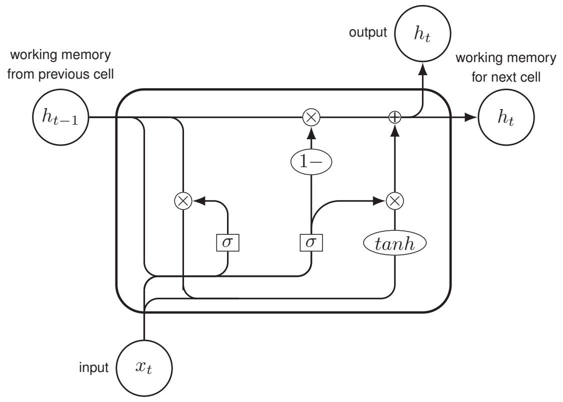
output
h_t
working memory from previous cell
h_{t-1}
working memory for next cell
h_t
$1-$
σ
σ
$tanh$
input
x_t

Figure 2: GRU Network Architecture

the literature such as Gated Recurrent Units (GRU). For the experiments in this paper, GRU architecture was used since it had shown accuracy comparable to LSTM but had better compute and memory performance.

A Gated Recurrent Unit (GRU) is a simplified architecture compared to LSTM (Cho et al. 2014). In GRU, fewer gates and only one kind of working memory h is used, thus making it computationally less expensive. The GRU cell has two inputs: input x at time t and memory h from time $t-1$ as shown in Fig. 2. The computations in a GRU cell are described below.

- ***Update gate*** defines how much of the memory should be saved and is computed as follows:
$$u(\cdot) = \sigma(w^{(ux)} \cdot x_t + w^{(uh)} \cdot h_{t-1} + b^{(u)})$$
- ***Reset gate*** determines if current state needs to be combined with memory from previous state and is computed as follows:
$$r(\cdot) = \sigma(w^{(rx)} \cdot x_t + w^{(rh)} \cdot h_{t-1} + b^{(r)})$$
- The output of reset gate is multiplied with output from previous state. This multiplied value, together with current input is then subjected to a non-linearity, mostly $tanh$ to produce the candidate memory. The candidate memory is computed as follows:
$$\tilde{h}(\cdot) = \tanh(w^{(\tilde{h}x)} \cdot x_t + w^{(\tilde{h}h)} \cdot (r_t \cdot h_{t-1}) + b^{(\tilde{h})})$$
- From the candidate memory, previous state output and update gate output, the final output is computed as follows:
$$h_t = (u_t \cdot \tilde{h}_t) + ((1 - u_t) \cdot h_{t-1})$$

Experiment and Results

The Dataset

The data for the experiments came from California Performance Measurement System. In this data set, real-time average speed data, aggregated at 5-minute intervals, from 88 detectors located at different highways across the state of

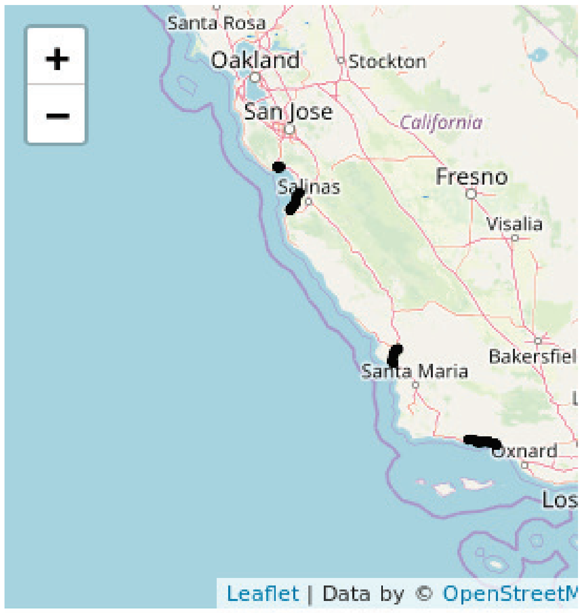

Figure 3: Location of all selected detectors in California

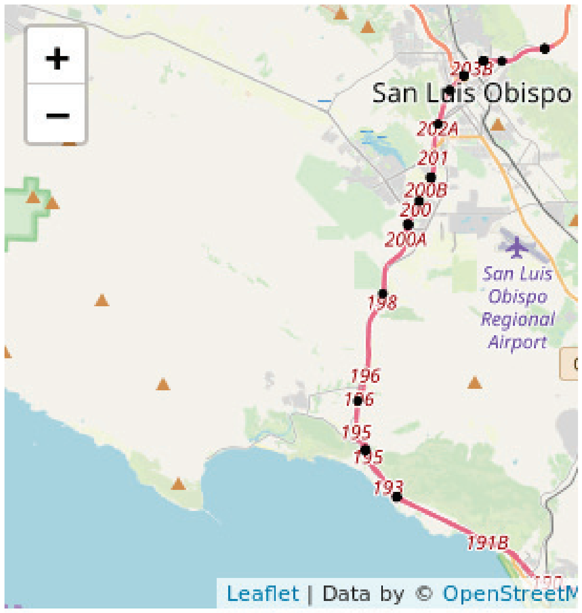
San Luis Obispo
203B
202A
201
200B
200
200A
198
196
196
195
195
193
191B
San Luis Obispo Regional Airport
Leaflet | Data by © OpenStreetM

Figure 4: Location of the selected detectors in San Luis Obispo area of California

California was collected at the locations shown in Fig. 3 and 4.

Collection of data from a variety of highways, freeways, and tollways ensures that we capture a diversity of different kinds of patterns. The western areas of California were selected because the roads in this district connect to different commercial areas from Silicon Valley to San Diego, thus making more data available for the purpose of research.

The data was filtered to include records only from years 2015 to 2018, from the third month to eighth month, and from weekdays Monday to Friday. All sensor data was scaled to be between 0 and 1, and transformed to window size of 24 steps (i.e. 2 hours) as features and 12 steps (i.e. 1 hour) as labels. No data was smoothed with statistical methods such as de-trending, de-seasonalizing, and differencing etc. because we wanted to avoid the effects that such pre-processing can have on the predictions.

Experimental Setup

Since GRU architectures were found to perform best in preliminary experiments, we focus our attention on GRU architecture in this paper. We consider GRU-based models with all the three strategies. The models were built using Keras, the popular high-level deep learning library that sits on top of the TensorFlow framework from Google (Fandango 2017; 2018). As the primary goal was to test the effect of augmenting the network architecture with the strategies, the best hyper parameter configuration from preliminary runs was picked for all the three strategies. These best hyper-parameters were obtained from the MIMO strategy.

A separate model was built for each station sensor identifier and strategy combination. For each model the experiment was repeated 10 times, and the mean of these 10 results was considered for further calculations. For the purpose of building the model, the first 80% data was used for training and the next 10% data was used for validation. The remaining 10% data was used as test set for reporting the results.

Performance Metrics

Four different metrics are popular in the traffic speed prediction community, hence, all of the following metrics were computed on the test data (Barros, Araujo, and Rossetti 2015):

- Mean Square Error (MSE) = $\frac{1}{N}\sum_{t=1}^{N}\left(\hat{y_t} - y_t\right)^2$
- Mean Absolute Error (MAE) = $\frac{1}{N}\sum_{t=1}^{N}\left|\hat{y_t} - y_t\right|$
- Mean Absolute Percentage Error (MAPE) = $\frac{100}{N}\sum_{t=1}^{N}\left|\frac{\hat{y_t}-y_t}{y_t}\right|$
- Symmetric MAPE (SMAPE)=$\frac{100}{N}\sum_{t=1}^{N}\left|\frac{(\hat{y_t}-y_t)\times 2}{\hat{y_t}+y_t}\right|$

Experiment Results

From the results we observe that — except for a couple of detectors out of the 88 — the recursive strategies almost always had the worst error for any of the error metric (Fig. 5). Thus the prediction error of recursive strategies was almost always higher as compared to direct and MIMO strategies. The plots of the performance metrics for the test data are shown in Fig. 5 and 6. MIMO strategies are basically RNN networks that can be further tuned to give better results, and these strategies performed reasonably well.

To confirm this visual inspection, a non-parametric Friedman rank-sum test was applied to the test data metrics to find if the difference between the strategies was significant. The Friedman test ranks the strategies from 1 to n for each

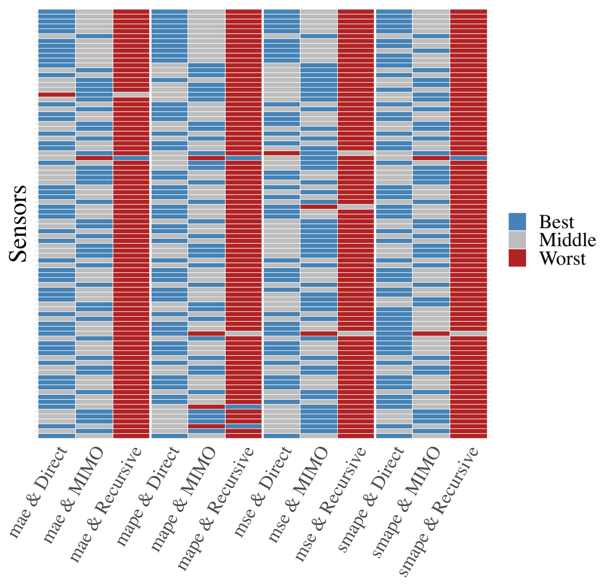

Figure 5: Error measures for all three strategies

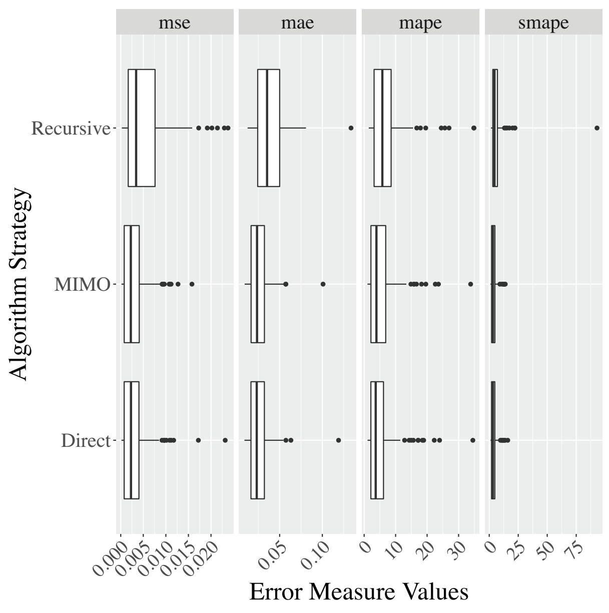

Figure 6: Boxplots for all error measures and strategies

dataset separately. After the ranking, the test compares the average rank of all strategies. The p-values of the Friedman test for MAE, MSE, MAPE, and SMAPE were 1.1e-27, 1.7e-27, 1.5e-25, 6.5e-28 respectively. The p-values of the Friedman test for all the four metrics were less than 0.05, thus the null hypothesis was rejected, and we conclude the differences are significant.

Shaffer's post-hoc test was applied to find out the if the pairwise differences were significant. From the results of Shaffer's post-hoc test, there was not enough evidence to reject the null hypothesis (that the strategies produce similar error values) between the direct and MIMO strategies. However, the direct strategy was 10 to 12 times more computationally expensive because of the need to build 12 separate models and to run inference 12 times for each of the prediction horizon time steps.

Conclusion and Future Work

In this paper we presented our results from augmenting recurrent neural networks with two popular strategies for multi-step-ahead predictions. Through our experiments we observed that recurrent strategies do not perform as well as MIMO and direct strategies.

In the recurrent strategy, first one-step-ahead is predicted and then the predicted value is combined with the observed data to predict the next time-step value. As we go farther in our prediction horizon, the input data consists of more predicted entries, thus the cumulative effect of the predicted error increases the final error metric values. That is why out of 88 datasets, the recurrent strategy had the lowest score for only a couple of datasets.

The MIMO and direct strategies produced almost comparable results, but direct strategies are computationally expensive. For predicting 12 times ahead, a direct strategy is 10 to 12 times slower and consumes 12 times more resources in order to build the 12 models. Thus we will focus our future research on MIMO models. Our observation aligns with similar findings in applying these strategies to classical machine learning models (Wen, Torkkola, and Narayanaswamy 2017; Taieb et al. 2012).

In the future we plan to experiment further on data from detectors on Florida D5 highways such as SR-417, SR-408 and I-4. We also plan to run these experiments with tunable synthetic data sets such as Mackey-Glass chaotic time series. Further, we would like to learn the difference between creating a global model for all the detectors vs. creating a separate model for each detector. Finally, we want to examine the effectiveness of transfer learning so that the models do not have to be trained fully on newly arrived data.

References

An, N. H., and Anh, D. T. 2015. Comparison of Strategies for Multi-step-Ahead Prediction of Time Series Using Neural Network. In *Proceedings of 2015 International Conference on Advanced Computing and Applications (ACOMP)*, 142–149. IEEE.

Barros, J.; Araujo, M.; and Rossetti, R. J. F. 2015. Short-term real-time traffic prediction methods: A survey. In *Proceedings of the International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 132–139. IEEE.

Bontempi, G.; Taieb, S. B.; Borgne, Y.-A. L. Y.-A.; Ben Taieb, S.; Borgne, Y.-A. L. Y.-A.; Taieb, S. B.; and Borgne, L. 2013. Machine Learning Strategies for Time Series Forecasting. *Business Intelligence* 138(September):62–77.

Chang, F.-J.; Chiang, Y.-M.; and Chang, L.-C. 2007. Multi-step-ahead neural networks for flood forecasting FI-. *Hydrological Sciences Journal* 52(1):114–130.

Chen, Y.-y.; Lv, Y.; Li, Z.; and Wang, F.-y. 2016. Long Short-Term Memory Model for Traffic Congestion Prediction with Online Open Data. In *Proceedings of the 19th IEEE Conference on Intelligent Transportation Systems (ITSC)*, 132–137. IEEE.

Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv*.

Dai, X.; Fu, R.; Lin, Y.; Li, L.; and Wang, F.-Y. 2017. DeepTrend: A Deep Hierarchical Neural Network for Traffic Flow Prediction. *arXiv*.

Dai, X.; Fu, R.; Zhao, E.; Zhang, Z.; Lin, Y.; Wang, F.-Y.; and Li, L. 2019. DeepTrend 2.0: A light-weighted multi-scale traffic prediction model using detrending. *Transportation Research Part C: Emerging Technologies* 103:142–157.

Elman, J. L. 1990. Finding Structure in Time. *Cognitive Science* 14(2):179–211.

Fandango, A., and Kapoor, A. 2018. Investigation of Iterative and Direct Strategies with Recurrent Neural Networks for Short-Term Traffic Flow Forecasting. In *Proceedings of the International Conference on Advances in Computing and Data Sciences (ICACDS)*.

Fandango, A., and Wiegand, R. P. 2018. Towards investigation of iterative strategy for data mining of short-term traffic flow with Recurrent Neural Networks. In *Proceedings of the 2nd International Conference on Information System and Data Mining (ICISDM)*.

Fandango, A. 2017. *Python Data Analysis, Second Edition*. Packt Publishing.

Fandango, A. 2018. *Mastering TensorFlow 1.x: Advanced machine learning and deep learning concepts using TensorFlow 1. x and Keras*. Packt Publishing.

Fu, R.; Zhang, Z.; and Li, L. 2016. Using LSTM and GRU neural network methods for traffic flow prediction. In *Proceedings of the 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 324–328. IEEE.

Hamzaçebi, C.; Akay, D.; and Kutay, F. 2009. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications* 36(2 PART 2):3839–3844.

Koesdwiady, A., and Karray, F. 2018. New Results on Multi-Step Traffic Flow Prediction. *arXiv*.

Koesdwiady, A.; Khatib, A. E.; and Karray, F. 2018. Methods to Improve Multi-Step Time Series Prediction. *Proceedings of the International Joint Conference on Neural Networks*.

Li, Y.; Wu, C.; and Yoshinaga, T. 2019. Traffic Flow Prediction with Compact Neural Networks. In *Proceedings of the 2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cy*, 1072–1076. IEEE.

Luo, X.; Li, D.; Yang, Y.; and Zhang, S. 2019. Spatiotemporal traffic flow prediction with KNN and LSTM. *Journal of Advanced Transportation* 2019.

Rahman, R., and Hasan, S. 2018. Short-Term Traffic Speed Prediction for Freeways during Hurricane Evacuation: A Deep Learning Approach. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems, Proceedings (ITSC)*, 1291–1296. IEEE.

Rahman, R. 2019. *Applications of Deep Learning Models for Traffic Prediction Problems*. Ph.D. Dissertation, University of Central Florida.

Shao, H., and Soong, B.-H. 2016. Traffic flow prediction with Long Short-Term Memory Networks (LSTMs). In *Proceedings of the IEEE Region 10 Conference (TENCON)*, 2990–2993. IEEE.

Taieb, S. B., and Hyndman, R. J. 2012. Recursive and direct multi-step forecasting : the best of both worlds.

Taieb, S. B.; Bontempi, G.; Atiya, A. F.; and Sorjamaa, A. 2012. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems With Applications* 39(8):7067–7083.

Taieb, S. B. 2014. *Machine Learning Strategies for multi-step-ahead Time Series Forecasting*. Ph.D. Dissertation, Université Libre de Bruxelles, Belgium.

Tian, Y., and Pan, L. 2015. Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network. In *Proceedings of the IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 153–158. IEEE.

Tian, Y.; Zhang, K.; Li, J.; Lin, X.; and Yang, B. 2018. LSTM-based traffic flow prediction with missing data. *Neurocomputing* 318:297–305.

Tibbitt, M. W.; Bae, H.; Dokmeci, M. R.; and Khademhosseini, A. 2013. Beyond One-Step-Ahead Forecasting: Evaluation of Alternative Multi-Step-Ahead Forecasting Models for Crude Oil Prices. *Energy Economics* 40:405–415.

Wen, R.; Torkkola, K.; and Narayanaswamy, B. 2017. A Multi-Horizon Quantile Recurrent Forecaster. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*.

Yang, B.; Sun, S.; Li, J.; Lin, X.; and Tian, Y. 2019. Traffic flow prediction using LSTM with feature enhancement. *Neurocomputing* 332:320–327.