Australasian Transport Research Forum 2022 Proceedings
28-30 September, Adelaide, Australia
Publication website: http://www.atrf.info

Deep-learning methods for long-term traffic flow forecasting

Xiao Zheng[1], Majid Sarvi[2], Saeed Asadi Bagloee[3]

[1, 2, 3]Faculty of Engineering and Information Technology, University of Melbourne, Melbourne, Australia

Email for correspondence: xzheng5@student.unimelb.edu.au

Abstract

Effective long-term forecasting of traffic flow has many applications for Intelligent Transportation Systems. With more historical measurements being available, data-driven methods are becoming promising for conducting this forecasting task. Existing research focuses on prediction for up to 1 day, yet there is an emerging demand for a longer forecasting horizon. This paper first reviews representative data-driven methods for traffic forecasting. Subsequently, the promising Sequence to Sequence (Seq2Seq) deep-learning methods are evaluated on the long-term forecasting up to 14 days ahead. Their performances on 1 day, 7 days and 14 days ahead traffic flow forecasting are compared and discussed. Then, the impacts of the shrinking size of training data and holiday traffic are investigated. Based on a real-world and large dataset from Melbourne, Australia, test results indicate that a state-of-the-art Transformer-based method, Informer, generates superior results than Seq2Seq RNN, LSTM, and GRU, especially when the forecasting span is extended. Besides, although the accuracy of Informer slightly degrades with the decreasing size of training data, 2 months of training data seem enough for it to produce a decent performance. In addition, the investigation of holiday traffic reveals its evident impact on forecasting accuracy.

1. Introduction

Intelligent Transportation System (ITS) aims to increase the operational efficiency and capacity of the transportation system by creating an integrated system of people, roads and vehicles (An et al., 2011). An efficient ITS environment requires a continuous flow of information regarding how traffic conditions evolve with time (Lieu, 2000), and one of the most frequently studied conditions is traffic flow (Vlahogianni et al., 2004), which refers to the number of vehicles passing through a given point on a road segment in a certain time span. Traffic flow forecasting can be used for specific tasks ranging from road condition controlling (Jiang and Adeli, 2005) to travel planning (Lee et al., 2009), hence is strongly needed for individual road users, business sectors, and government agencies.

A predominant change in ITS recently is extensive data can be collected from various sources (Zhang et al., 2011), which prompts the prevalence of data-driven methods for traffic forecasting. Unlike knowledge-driven methods employing analytical or simulation models (Cascetta, 2013), data-driven approaches develop models directly learning the traffic dynamics from traffic data, and are generally more accurate and robust (Van Lint and Van Hinsbergen, 2012). From the perspective of the forecasting period, data-driven methods can be classified into short-term (from a few seconds to a few hours) (Vlahogianni et al., 2014) and long-term (longer than short-term and up to 24 hours) forecasting (Hou et al., 2015), and the former has attracted most effort till several years ago (Vlahogianni et al., 2014). More recently, research

applying deep-learning methods, an advanced category of data-driven methods, to traffic forecasting (such as DCRNN (Li et al., 2017), ASTGCN (Guo et al., 2019) and GMAN (Zheng et al., 2020)) is mainly evaluated over a 5-minutes to 24-hours prediction window. Compared to some knowledge-driven methods that can generate traffic prediction for years ahead, the studied forecasting horizon is limited for data-driven methods, creating a noticeable research gap. Effective traffic forecasting for a longer term is needed since it enables action optimization at more steps in the future. For authority, this means more reaction time to traffic conditions; for traveller, this allows them to plan further ahead.

This paper investigates the performance of promising data-driven methods on a longer term (up to 14 days) traffic flow forecasting. The study is based on data collected from a single location, which applies to a common scenario in which only one detector is deployed, or the detectors are located too far to generate any spatial dependency. Representative works of data-driven methods for traffic forecasting are reviewed first. Then, promising Sequence to Sequence (Seq2Seq) methods, including Recurrent Neural Network (RNN), Long Short Term Memory Network (LSTM), Gated Recurrent Unit Network (GRU) and a state-of-the-art method, Informer, are evaluated with real-world signalized arterial data to study their performance in different scenarios including the increased forecasting horizon and the reduced size of the training data. Lastly, the impact of holiday traffic on forecasting accuracy is investigated.

2. Review of related work

Data-driven approaches for traffic forecasting develop models directly learning the traffic dynamics from traffic data, and can be classified into statistical methods, machine learning methods, and deep learning methods (Tedjopurnomo et al., 2020).

Statistical methods build a data-driven statistical model and are typically applied to short-term (commonly 1 step ahead) forecasting. The most representative work is the Auto-Regressive Integrated Moving Average (ARIMA) (Box et al., 1970) identifying recurring patterns from historical data. One modification of it, Seasonal ARIMA, has been applied to 1 step (15 minutes) ahead traffic forecasting (Williams and Hoel, 2003). Another influential method is the Kalman Filter (KF), a recursive minimum variance data processing algorithm (Kalman, 1960). Methods based on KF were proven to be an efficient method for short term traffic forecasting (Okutani and Stephanedes, 1984, Jin et al., 2013, Guo et al., 2014, Emami et al., 2019, Emami et al., 2020). Although with good interpretability, statistical methods are typically designed for small datasets and require uninterrupted data. Another major drawback is they usually assume specific properties of the data (for example, stationarity), whereas traffic data is often too dynamic and complex to satisfy these assumptions.

Machine learning methods learn from data by assembling mathematical models (Bagloee et al., 2018), which predict traffic typically by solving the corresponding regression problem. For instance, Support Vector Regression transfers a non-linear regression problem to a linear one by using a non-linear mapping to map data into a high-dimensional feature space (Müller et al., 1997), and has been applied to 15-minute ahead traffic forecasting (Wang et al., 2015). Another example is K-nearest Neighbours method based on the assumption that observations which are close in feature-space are likely to be more similar (Devijver and Kittler, 1982), which has been used to predict up to 5 time steps into the future (Zhang et al., 2013, Yu et al., 2016a). Although machine learning methods are proven to be able to model non-linearity and extract some complex correlations in the traffic data, these approaches either rely on strong stationary assumptions of the data or not being able to consider highly non-linear temporal dependency (Yu et al., 2016b).

Deep learning is a part of the machine learning method and a thriving field today with increased data size and computing power (Liu et al., 2020). Deep learning methods use multiple-layer architectures to extract inherent features in data and map the input traffic data directly to the required output. They are believed to have the potential to find out highly non-linear temporal dependency and have acknowledged capability of modelling complex traffic patterns (Tedjopurnomo et al., 2020). The mainstream types of deep-learning models applied to the traffic forecasting problem considering exclusively temporal dependency include RNN and its variants and the Transformer-based models (Tedjopurnomo et al., 2020).

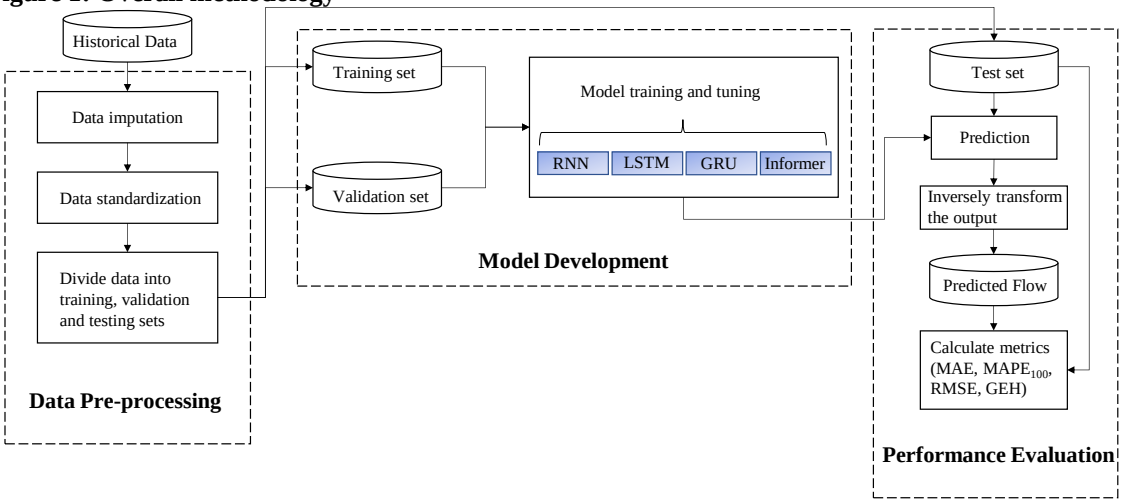
RNN has the ability to store information, thus allowing it to capture dependencies between different parts of the sequence (Schmidt, 2019). It is often visualized as a chain of nodes, for each node, the input from the corresponding time step and a summary of information from all previous time steps is fused and passed. However, when processing a sequence of significant length, RNN suffers from gradient vanishing (Hochreiter, 1998). LSTM was proposed as a variance of RNN to solve this issue. Its gated sophisticated structure enhances the model's ability to learn and memorize long dependency features (Hochreiter and Schmidhuber, 1997). Moreover, a simplified version of LSTM is GRU, which contains fewer gates (Cho et al., 2014). A Seq2Seq architecture is later introduced to allow RNN-based methods to predict an output sequence with arbitrary length (Sutskever et al., 2014). This architecture works by encoding the input information into a vector in the encoder and sending this vector into the decoder where the output is generated. Notably, a method based on Seq2Seq LSTM demonstrated promising results for as long as 1 day (288 steps ahead) traffic forecasting (Wang et al., 2020).

The attention mechanism is introduced to Seq2Seq RNNs for dealing with longer sequences (Bahdanau et al., 2014). Intuitively, attention is a weighted combination of a sequence of data, where the weights are determined by the level of similarity among data points. The attention mechanism allows the model to search among every historical time step in the encoder, then decide their importance and select the relevant information for predicting in the decoding procedure. Completely avoiding recurrent structure, Transformer (Vaswani et al., 2017) is a recent model utilizing an attention mechanism for different time steps in a single sequence, namely, self-attention (Cheng et al., 2016), leading to improved performance on capturing long-term dependency. Vanilla Transformer contains an encoder and a decoder, each being a stack of several identical blocks. Each encoder block contains a multi-head self-attention module and a position-wise feed-forward network, and employs residual connection (He et al., 2016) and layer normalization (Ba et al., 2016). Decoder blocks additionally insert cross-attention modules between encoder and decoder. When applying Transformer to perform long-term forecasting, one of the major issues is the T-quadratic computation and memory consumption on T length inputs/outputs for a single layer. Informer (Zhou et al., 2020) is a successful attempt to improve the efficiency of the Transformer, while maintaining accuracy.

3. Methodology

Since deep-learning methods have shown great potential in long-term traffic forecasting from the review of related work, four major Seq2Seq deep learning models, including RNN and its two most acknowledged variances, LSTM and GRU, as well as the Informer, the state-of-the-art Transformer-based model for long sequence time-series forecasting, are selected for evaluation with a real-world urban arterial traffic dataset. This section elaborates on these methods, with the overall methodology illustrated in Figure 1.

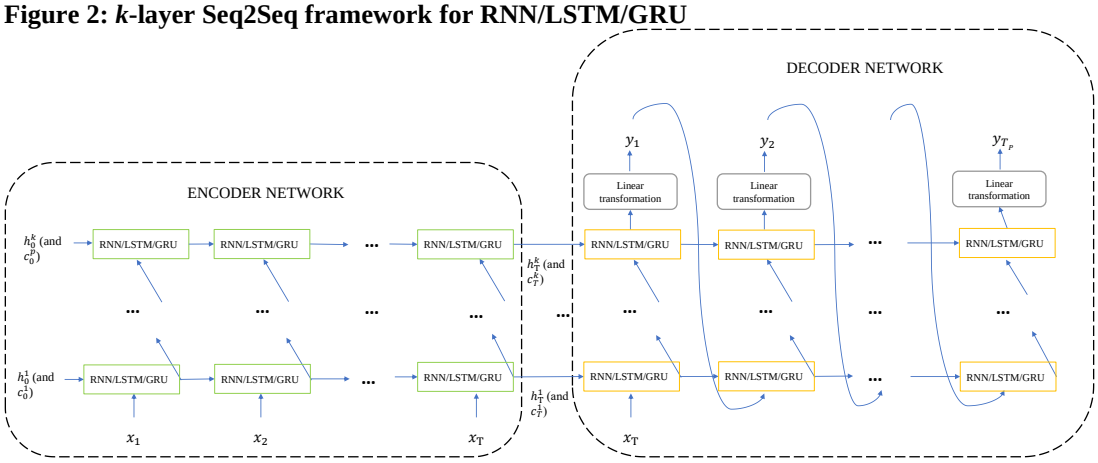
Figure 1: Overall methodology



3.1. RNN, LSTM and GRU

When predicting traffic using RNN and its variances, we introduce a multilayer Seq2Seq framework illustrated in Figure 2. As shown in Figure 2, the Seq2Seq model is composed of two stacked RNN/LSTM/GRU networks named encoder and decoder, respectively. The input sequence is encoded into semantic vectors, which are then decoded in the decoder to generate the output sequence. In addition, each RNN/LSTM/GRU network stacks a series of layers, which can help capture higher-level representations of sequence data (LeCun et al., 2015). Note that a dropout (Srivastava et al., 2014) layer is added to the outputs of each layer except the last layer. As a powerful regularization technique, dropout is a process of randomly setting elements of the outputs to 0 with a designed probability p, and then scaling the resulting outputs by a factor of $\frac{1}{1-p}$.

Figure 2: *k*-layer Seq2Seq framework for RNN/LSTM/GRU



In the m layer of the encoder, a sequence of T length is input into the model, and each element of this sequence, x_t^m, contains the information of a corresponding time-step. A single layer of RNN can be viewed as a list of nodes, and for each node, a hidden state, h_t^m, is calculated with the following equation:

$$h_t^m = \tanh\left(W_i^m x_t^m + b_i^m + W_h^m h_{(t-1)}^m + b_h^m\right) \quad 1)$$

where W_i^m, W_h^m, b_i^m and b_h^m are trainable parameters, and $h_{(t-1)}^m$ is the hidden state from the last node, or the given initial value h_0^m. The generated hidden states $[h_1^m, h_2^m, \cdots, h_T^m]$ after dropout are used as the input of the subsequent layer, that is, $x_t^{m+1} = h_t^m$. Suppose there are k layers in total, the encoder will generate $[h_T^1, h_T^2, \cdots, h_T^k]$, which, together with x_T, is then sent into the decoder. In the decoder, they are used to calculate hidden states for each layer (for instance, h_{T+1}^m) with Equation 1) and the hidden states passing between the layers in the same way. The calculated h_{T+1}^k then goes through a linear transformation to obtain the first forecasting value, y_1. y_1 is then used as the input of the second node in the first layer. Following the same procedure, the forecasting of every time step after the input sequence is calculated recursively, this output sequence ends with y_{T_p}, when the task is predicting T_p time steps.

Compared to the multilayer Seq2Seq RNN framework, the main change in the corresponding LSTM network is the computation in the nodes. Each node of LSTM receives the corresponding input x_t, a hidden state $h_{(t-1)}$ and a hidden memory cell state, c_{t-1}, and maintains three gates including a forget gate f_t, a input gate i_t, and a output gate o_t. The memory cell and gates can help LSTM learn long-term dependencies by effectively storing and passing useful historical information. The calculation is performed using the following equations (for notation simplicity, subscript m is omitted hereinafter):

$$f_t = S\left(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}\right) \quad 2)$$

$$i_t = S\left(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}\right) \quad 3)$$

$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}\right) \quad 4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad 5)$$

$$o_t = S\left(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}\right) \quad 6)$$

$$h_t = o_t \odot \tanh(c_t) \quad 7)$$

where S is the sigmoid function, $\odot$ is the Hadamard product, W_{if}, W_{hf}, W_{ii}, W_{hi}, W_{ig}, W_{hg}, W_{io}, W_{ho} and b_{if}, b_{hf}, b_{ii}, b_{hi}, b_{ig}, b_{hg}, b_{io}, b_{ho} are trainable parameters. After dropout, h_t is passed to the corresponding subsequent layer as input.

Containing two gates only, GRU is a lighter version of LSTM. The studied multilayer Seq2Seq GRU shares the same architecture with the abovementioned multilayer Seq2Seq RNN, while the only difference is the calculation taking place in each node, which is now:

$$r_t = S(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \quad 8)$$

$$z_t = S(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \quad 9)$$

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hg}h_{(t-1)} + b_{hn})) \quad 10)$$

$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{(t-1)} \quad 11)$$

where S is the sigmoid function, $\odot$ is the Hadamard product, W_{ir}, W_{hr}, W_{iz}, W_{hz}, W_{in}, W_{hg}, and b_{ir}, b_{hr}, b_{iz}, b_{hz}, b_{in}, b_{hn}, are trainable parameters.

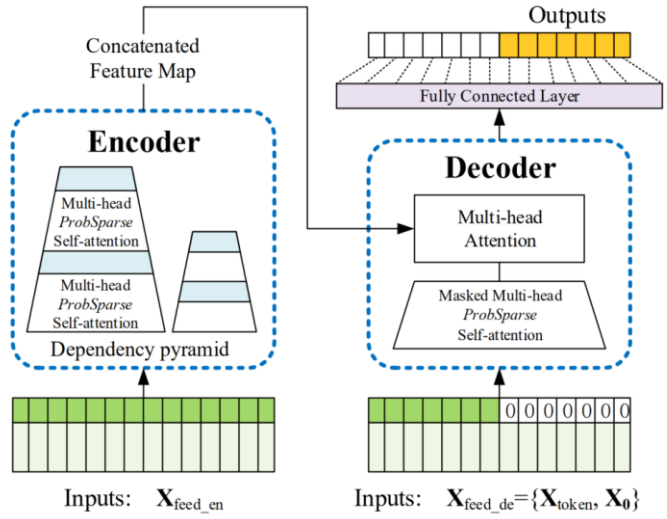
3.2. Informer

Informer (Zhou et al., 2020) is developed to alleviate the computation cost of the vanilla Transformer. The *ProbSparse* Self-attention mechanism proposed in Informer selects and

keeps only dominant dot-product pairs in attention calculation by an approximation of the Kullback-Leibler divergence. It also uses a generative style decoder, which generates the production in one forward operation, to replace the step-by-step decoding procedure in the vanilla Transformer. This allows fast predicting procedure and avoids error cumulation during decoding.

As demonstrated in Figure 3, in the encoder of Informer, the data sequence with embedded time stamp and position stamp, X_{feed_en}, is put into the Multi-head *ProbSparse* self-attention layer, where for each head, the queries Q, keys K and values V are firstly calculated by different linear projections of the input. Then, we calculate $U = c \times \text{celi}(\ln(L_k))$ and $u = c \times \text{celi}(\ln(L_Q))$, where c is a hyperparameter, and L_k and L_Q is the length of the sequence of keys and queries, respectively. Next, we randomly select U dot-product pairs from K as $\overline{K}$, and time $\overline{K}$ by Q to get the sample score S. The following steps involve computing M as the difference between maximum S and mean S and selecting the top-u M and getting their corresponding indices to locate $\bar{Q}$. The final output is computed by partially updating the average value of V with $\text{softmax}(\bar{Q}K^T/\sqrt{d}) \cdot V$ (d is the input dimension). The output goes through a 'dropout, add and layer norm' process. This is to say, we perform dropout to the output, add it to the input of the layer, and then perform a layer normalization to the sum. The result then goes through a fully connected layer and the same 'dropout, add and layer norm' process. Moreover, Informer puts the output through a self-attention distilling process, which involves a down convolutional transformation, then a batch normalization (Ioffe and Szegedy, 2015), an activation with Elu (Clevert et al., 2015), and a Max-Pooling (Giusti et al., 2013). The result will then be used as the input of the next encoder layer, and the total memory usage for multiple layers is restrained. To enhance the robustness, Informer also allows building halving replicas of the main stack, letting it go through a similar operation, and concatenating the outputs together.

Figure 3: Overall structure of the Informer (Zhou et al., 2020)



A concatenation of an earlier time slice before the predicting sequence named 'decoder start token', X_{token}, and the place holder for the predicting sequence, X_0, which is summed with a similarly embedded time stamp and position stamp, is then sent into the decoder. This input can go through another Multi-head *ProbSparse* self-attention layer. Here, an additional 'mask mechanism' is applied to prevent a current position from attending to subsequent positions by only allowing a certain query to visit the keys and values before or at the position of this query.

The output will then go through a 'dropout, add and layer norm' process as before, and then be put into a following cross-attention layer, where it is transformed into queries, and the output from the encoder is transformed to keys and values. Perform 'dropout, add and layer norm' and send the outcome through a fully connected layer, then perform 'dropout, add and layer norm' again. This outcome becomes the input of the next layer of the decoder. At the end of all decoder layers, the results will go through a layer normalization and be projected through a linear model to get the forecasting results.

4. Performance evaluation

In this section, Informer and the proposed multilayer Seq2Seq RNN, LSTM and GRU are evaluated with real-world signalized arterial data to investigate their performance in predicting the long-term traffic flow.

4.1. Datasets

Three flow datasets from AIMES (http://aimes.com.au/) are used for evaluation. Each dataset is collected from an arterial segment in the Melbourne urban area, whose location is shown in Figure 4. Data from each segment is treated separately to produce individual models, neglecting any potential spatial correlation. Each dataset contains a whole year traffic flow of 2019, with 3 days of data missing. Considering the small proportion of missing data (0.82%), the imputation is conducted simply by replacing them with the data from the same period last week. The data granularity is 15 minutes. Some statistical properties of the datasets after imputation are summarized in Table 1.

Figure 4: Location of the dataset road segments (marked and numbered in blue)

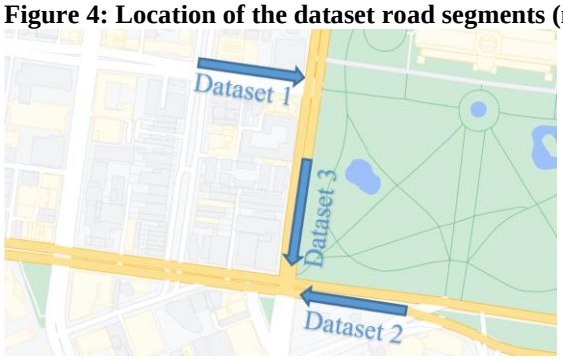
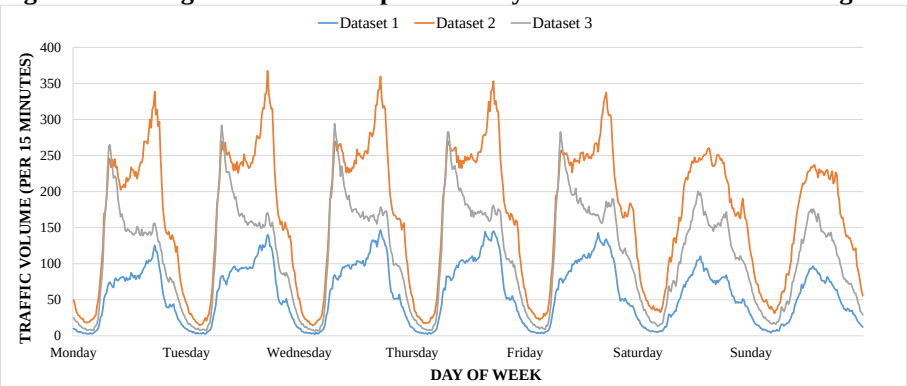


Table 1: Statistical properties of flow data

Dataset No.	Statistical properties (no of vehicle/15 mins)					
	Mean	Std	Percentile			
			25^{th}	50^{th}	75^{th}	99^{th}
1	60	46	16	54	95	168
2	169	98	70	180	250	361
3	112	75	40	113	168	295

Average traffic flow per each day of the week is calculated from the first 8 months of data, as shown in Figure 5. It can be observed that on top of different daily patterns (for example, Dataset 3 has strong morning peaks and subtle afternoon peaks for weekdays, and the weekend traffic peaks before noon), all three datasets have an obvious weekly pattern being an overall decrease of flow and change of daily pattern during weekends, compared to the weekday traffic.

Figure 5: Average traffic volume per each day of the week over the first eight months



The size of the training/validation/testing dataset is 8 months/1 month/3 months, respectively. The flow data has been scaled by standardization with the statistics from the training set before being loaded into the models with a rolling stride of 1. Prior to metrics calculation, the predicted values from the models are inversely transformed.

4.2. Evaluation metrics

In this research, the widely used and reported metrics in academia, including Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), are utilized for performance evaluation. Especially, inspired by (Lippi et al., 2013), we measure Mean Absolute Percentage Error (MAPE) only over the results with corresponding ground truth >100 vehicles/15 minutes, namely, $MAPE_{100}$. This is to eliminate the disproportionate impact generated from the very low values. An additional metric used in this investigation is GEH statistics, which is an empirical calculation widely used in the industry to represent the goodness-of-fit of a model, and the formula is:

$$GEH = \sqrt{\frac{2(M-C)^2}{M+C}} \qquad 12)$$

where M is the hourly traffic flow from model forecasting and C is the corresponding ground truth. As the outputs from models are 15-minute level forecasting, 4 consecutive results are summed up and considered hourly traffic. An estimate generating a GEH index of 5.0 or less is deemed acceptable, while any GEH calculation greater than 10 is deemed to be unacceptable, and results in between would require attention and scrutiny (VicRoads, 2019). The use of GEH statistic will help understand if the performance is satisfying in the real-world application for the traffic forecasting problem.

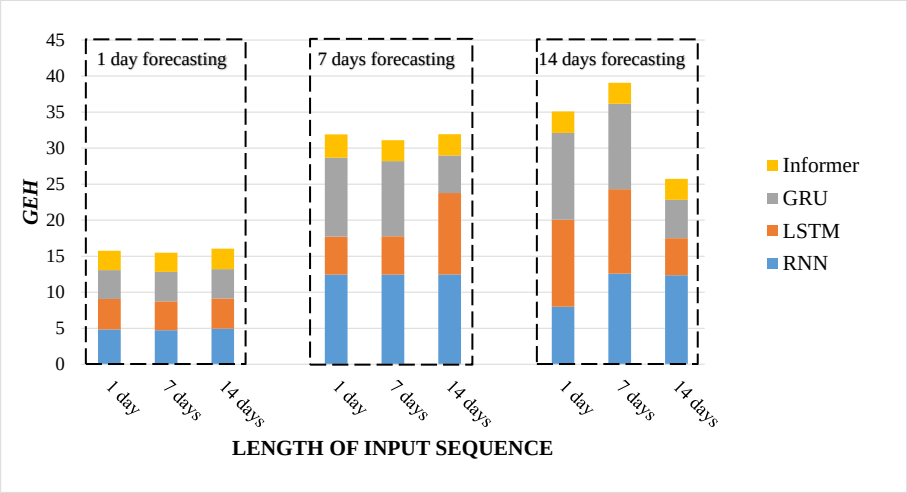
4.3. Implementation details

Experiments are performed with the following implementation details (key parameters are tuned on the held-out validation set):

- For Seq2Seq RNN, GRU and LSTM, both the encoder and decoder contain 3 layers (chosen from {1, 3, 6} layers), and the size of the hidden state and cell state is set as 512. Models are trained with batch size 32 and a loss function MSE. For the dropout layer, $p = 0.1$. The initial learning rate is 0.0001 (selected from {0.01,0.001,0.0001}), which is then reduced by multiplying $0.5^{no.of\ last\ epoch-1}$. Adam optimizer (Kingma and Ba, 2014) is used. Early stopping is performed in the way that if the validation loss increases in 3 consecutive epochs, the training is terminated.

- For Informer, the parameters in the originally proposed model (Zhou et al., 2020) are adopted, except that we adopted *probsparse* attention and its corresponding settings for both encoders and decoders. The initial value of the learning rate is tuned to be 0.0001 (selected from {0.01,0.001,0.0001, 0.00001}). The batch size is set as 2. The same loss function, learning rate decay strategy, optimizer and early stopping strategy used for Seq2Seq RNN/GRU/LSTM, is also adopted for Informer. The length of 'decoder start token' is set as 288 when the input length is 7 days and 14 days, and is set as 48 when the input length is 1 day.
- To identify the optimum length of input data for different prediction horizons, a group of experiments are conducted on Dataset 3, and the results are summarized in Figure 6. We, therefore, select the input length with the lowest GEH summed among all methods: for 1 day/7 days/14 days forecasting, the length of input data sequence is determined as 7 days/7 days/14 days, respectively. Meanwhile, preliminary observation shows that the general effect of input length on prediction accuracy is not obvious unless for a longer prediction horizon like 14 days. Further discussions of this will be conducted in future works.

Figure 6: Comparison of the impact of different input lengths on the results of different prediction horizons



4.4. Results and discussion

In this section, we compare the performance of the select methods and discuss the performance with the shrinking training data and the effect of holiday traffic.

4.4.1. Comparison of performance among the methods

Table 2 summarizes the evaluation results of traffic flow forecasting over the next 1 day (96 steps), 7 days (672 steps), and 14 days (1344 steps), with the best results marked in bold. According to Table 2, the results of RNN become unacceptable (GEH>10) when the prediction period is increased to 14 days. One primary reason is that RNN suffers from gradient vanishing and, therefore, only has a limited ability to learn information from a long time ago. Comparatively, LSTM and GRU can sometimes generate decent accuracy for 14-day forecasting, suggesting their superiority in dealing with longer sequences due to their gated/cell state mechanism. Informer achieves the best accuracy in all prediction tasks, and its superiority generally becomes more evident as the forecasting period prolongs, since the performance of other methods typically decays with the increase of the forecasting span. Remarkably, this accuracy decay is not evident for Informer. A key factor contributing to the improved

forecasting capacity of Informer might be its adopted Transformer framework (particularly the self-attention mechanism), which allows the model to review the entire input sequence and utilise the most relevant information, regardless of the length of the data sequence. Another main reason can be, for Informer, the whole prediction sequence is generated at once with its proposed generative decoding process, avoiding potential error accumulation from the recursive decoding process used by RNN/LSTM/GRU.

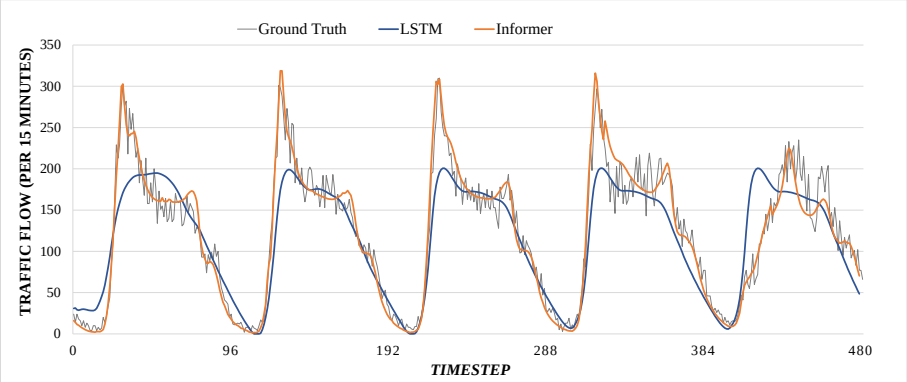
Table 2: Performance comparison of different methods with increasing forecasting span

Dataset No.	Forecasting Horizon	Metric	RNN	LSTM	GRU	Informer
1	1 day	MAE	16.79	15.33	14.99	**13.80**
		RMSE	23.93	22.56	21.85	**21.03**
		$MAPE_{100}$	21.54%	20.85%	20.13%	**18.13%**
		GEH	3.92	3.64	3.57	**3.00**
	7 days	MAE	23.00	40.42	21.24	**14.45**
		RMSE	31.03	48.85	29.07	**22.35**
		$MAPE_{100}$	28.29%	49.92%	25.59%	**17.98%**
		GEH	5.62	10.42	5.16	**3.36**
	14 days	MAE	43.85	41.94	20.91	**14.78**
		RMSE	51.91	50.27	28.78	**22.36**
		$MAPE_{100}$	53.46%	51.38%	25.55%	**19.62%**
		GEH	11.24	10.77	4.88	**3.41**
2	1 day	MAE	34.11	25.52	28.25	**23.6**
		RMSE	46.54	36.67	38.36	**36.1**
		$MAPE_{100}$	19.19%	14.45%	15.87%	**13.67%**
		GEH	5.12	3.68	4.28	**3.42**
	7 days	MAE	83.29	37.19	33.63	**23.64**
		RMSE	96.17	49.95	44.85	**37.11**
		$MAPE_{100}$	28.10%	20.92%	19.08%	**13.78%**
		GEH	13.37	5.65	5.05	**3.34**
	14 days	MAE	83.19	78.9	78.83	**22.78**
		RMSE	96.14	92.84	92.84	**34.53**
		$MAPE_{100}$	28.00%	26.95%	27.06%	**12.95%**
		GEH	13.3	12.6	12.57	**3.26**
3	1 day	MAE	26.4	21.93	22.37	**15.62**
		RMSE	39.44	32.65	33.26	**25.19**
		$MAPE_{100}$	18.53%	15.56%	15.48%	**11.72%**
		GEH	4.73	4.00	4.08	**2.67**
	7 days	MAE	63.4	29.1	53.04	**17.14**
		RMSE	75.16	42.24	63.71	**28.43**
		$MAPE_{100}$	30.59%	18.30%	28.64%	**12.58%**
		GEH	12.45	5.32	10.42	**2.9**
	14 days	MAE	63.2	28.43	29.3	**16.57**
		RMSE	75.22	41.45	42.51	**26.22**
		$MAPE_{100}$	30.32%	18.02%	18.88%	**12.29%**
		GEH	12.35	5.15	5.31	**2.93**

We present an example visualized result for 14-day forecasting in Figure 7. The result illustrated in Figure 7 is a slice of the prediction for Dataset 3 from 0:00 1st to 23:45 5th October 2019, which is the first 5 days of a single 14-day forecasting, with Informer and LSTM. As shown in Figure 7, Informer can capture different daily patterns on weekdays (from the beginning to time step 384) and weekends (from time step 385 to the end) with satisfying accuracy. Comparatively, the second-best method for this forecasting task, LSTM, only

captures a roughly daily pattern (higher flow in the morning) and cannot reproduce the weekly pattern (the difference between weekdays and weekends). Besides, after approximately the first 96 steps of forecasting, the rest of the prediction of LSTM seems only to repeat a similar pattern, which suggests that it cannot effectively preserve and pass useful information when the sequence becomes excessively long.

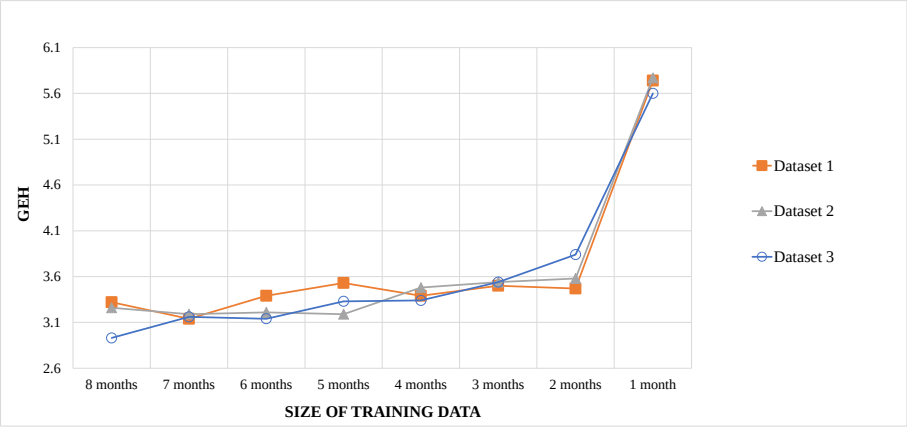
Figure 7: Visualized prediction for the first 5 days (480 steps) of a 14-day (1344 steps) forecasting with Informer and LSTM on Dataset 3



4.4.2. Influence of the size of training data

A common challenge in practice is that the amount of available data is often limited. And an ideal method should be able to produce stable accuracy when the size of data is reduced. Since Informer has demonstrated the best accuracy among the evaluated deep learning methods in this study, we further assess its performance on 14-day forecasting by gradually reducing the amount of training data while fixing the validation and test set. Figure 8 shows the change in GEH when the size of the training set is altered from 8 months to 1 month. Although a general increase in GEH can be observed as the size of training data reduces, no evident worsening of performance is noticed if 2 months of data can be used for training, which gives a preliminary indication of the required amount of training data to assure a satisfactory performance for 14-day forecasting. On the other hand, this may also imply that the studied Informer model did not effectively exploit more information from up to 8 months of the training set, for instance, other potential higher level seasonal patterns (such as monthly patterns).

Figure 8: The performance of Informer when the size of training data is reduced



4.4.3. Influence of the holiday traffic

Abnormal traffic patterns can be observed in the ground truth during the holiday period. As one example shown in Figure 9, for Dataset 3, traffic flow on 25th December (marked between red lines), which is a weekday and also the Christmas Day, does not exhibit a strong morning peak or a subtle afternoon peak like other non-holiday weekdays.

We further investigate the impact of holiday traffic on flow forecasting. An example prediction of Dataset 3 with Informer, covering the 'Christmas season', is displayed in Figure 9, from which it is evident that Informer still generates typical weekday and weekend predictions for the period with abnormal holiday traffic patterns, which can significantly jeopardize the forecasting accuracy. In Figure 10, we compare the results of 14-day Informer forecasting between the results covering all types of traffic conditions and the results calculated when holiday traffic forecasting is not considered. In the test set, holiday traffic is observed on 5th November (the 'Melbourne Cup') and from 23rd December to the end of the year (recognized as the 'Christmas season'). Results indicate that when removing the impacts of holiday traffic forecasting, the GEH can be lowered by an average of 0.37.

Figure 9: Visualized prediction for a 14-day forecasting with Informer on Dataset 3, covering a period of holiday traffic

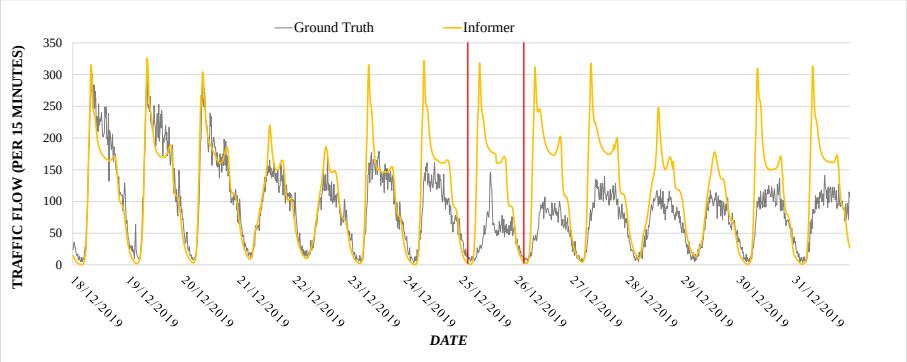
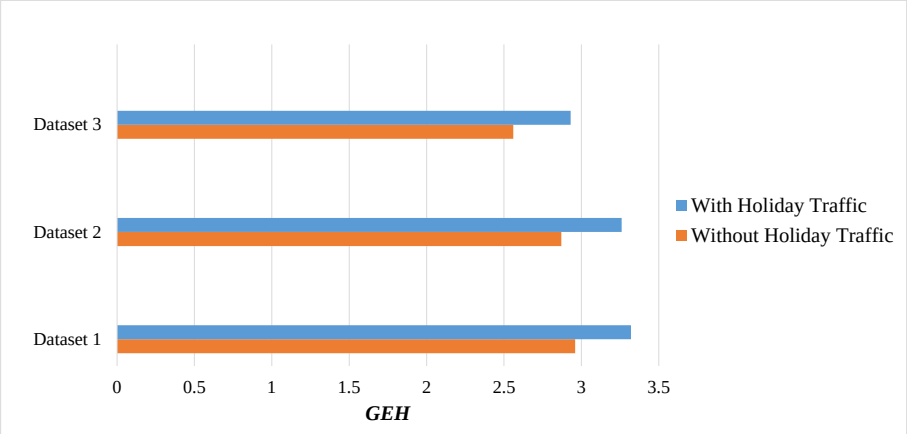


Figure 10: Performance comparison for 14-day forecasting with Informer when the accuracy calculation considers both non-holiday and holiday traffic, and considers exclusively non-holiday traffic



5. Conclusions

Accurate traffic flow forecasting is of great importance for ITS, and with more data being collected nowadays, data-driven methods have shown great potential in traffic forecasting

problems. Most efforts have gone to forecasting up to 1 day. On the other hand, longer-term prediction is required for its own applications. In this paper, after reviewing the representative data-driven methods, the performance of promising Seq2Seq deep learning forecasting methods (RNN, LSTM, GRU and Informer) when predicting traffic flow for up to 14 days is evaluated with real-world data collected from signalized arterials. Results indicate that Informer has strong performance improvements over other deep-learning methods being tested when dealing with an excessively long forecasting span, and the improvements become more predominant as the forecasting horizon increases. This highlights the effectiveness of a list of innovations of Informer, including its adopted Transformer framework and the proposed generative inference, both of which are well-suited for long-term traffic forecasting. Besides, although there is a slight accuracy decay when reducing the size of training data, 2 months of training data seems enough for Informer to perform well for 14-day forecasting. Lastly, holiday traffic has been found to adversely affect forecasting accuracy, making it an important question for future research. Potential solutions can be increasing the size of training data to provide enough instances for the model to learn this particular variation or embedding representative features into the model. Future work will also further investigate the impact of input length on prediction accuracy and the approaches to modelling spatial correlations between different road segments.

6. Acknowledgements

This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

References

AN, S.-H., LEE, B.-H. & SHIN, D.-R. A survey of intelligent transportation systems. 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, 2011. IEEE, 332-337.

BA, J. L., KIROS, J. R. & HINTON, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

BAGLOEE, S. A., ASADI, M., SARVI, M. & PATRIKSSON, M. 2018. A hybrid machine-learning and optimization method to solve bi-level problems. *Expert Systems with Applications,* 95**,** 142-152.

BAHDANAU, D., CHO, K. & BENGIO, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

BOX, G. E., JENKINS, G. M. & REINSEL, G. 1970. Time series analysis: forecasting and control Holden-day San Francisco. *BoxTime Series Analysis: Forecasting and Control Holden Day1970*.

CASCETTA, E. 2013. *Transportation systems engineering: theory and methods*, Springer Science & Business Media.

CHENG, J., DONG, L. & LAPATA, M. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H. & BENGIO, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

CLEVERT, D.-A., UNTERTHINER, T. & HOCHREITER, S. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

DEVIJVER, P. A. & KITTLER, J. 1982. *Pattern recognition: A statistical approach*, Prentice hall.

EMAMI, A., SARVI, M. & ASADI BAGLOEE, S. 2019. Using Kalman filter algorithm for short-term traffic flow prediction in a connected vehicle environment. *Journal of Modern Transportation,* 27**,** 222-232.

EMAMI, A., SARVI, M. & BAGLOEE, S. A. 2020. Short-term traffic flow prediction based on faded memory Kalman Filter fusing data from connected vehicles and Bluetooth sensors. *Simulation Modelling Practice and Theory,* 102**,** 102025.

GIUSTI, A., CIREŞAN, D. C., MASCI, J., GAMBARDELLA, L. M. & SCHMIDHUBER, J. Fast image scanning with deep max-pooling convolutional neural networks. 2013 IEEE International Conference on Image Processing, 2013. IEEE, 4034-4038.

GUO, J., HUANG, W. & WILLIAMS, B. M. 2014. Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies,* 43**,** 50-64.

GUO, S., LIN, Y., FENG, N., SONG, C. & WAN, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 2019. 922-929.

HE, K., ZHANG, X., REN, S. & SUN, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

HOCHREITER, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* 6**,** 107-116.

HOCHREITER, S. & SCHMIDHUBER, J. 1997. Long short-term memory. *Neural computation,* 9**,** 1735-1780.

HOU, Y., EDARA, P. & SUN, C. 2015. Traffic Flow Forecasting for Urban Work Zones. *IEEE Transactions on Intelligent Transportation Systems,* 16**,** 1761-1770.

IOFFE, S. & SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning, 2015. PMLR, 448-456.

JIANG, X. & ADELI, H. 2005. Dynamic wavelet neural network model for traffic flow forecasting. *Journal of transportation engineering,* 131**,** 771-779.

JIN, S., WANG, D.-H., XU, C. & MA, D.-F. 2013. Short-term traffic safety forecasting using Gaussian mixture model and Kalman filter. *Journal of Zhejiang University SCIENCE A,* 14**,** 231-243.

KALMAN, R. E. 1960. A new approach to linear filtering and prediction problems.

KINGMA, D. P. & BA, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LECUN, Y., BENGIO, Y. & HINTON, G. 2015. Deep learning. *nature,* 521**,** 436-444.

LEE, W.-H., TSENG, S.-S. & TSAI, S.-H. 2009. A knowledge based real-time travel time prediction system for urban network. *Expert systems with Applications,* 36**,** 4239-4247.

LI, Y., YU, R., SHAHABI, C. & LIU, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

LIEU, H. C. 2000. Traffic estimation and prediction system.

LIPPI, M., BERTINI, M. & FRASCONI, P. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems,* 14**,** 871-882.

LIU, J., LI, T., XIE, P., DU, S., TENG, F. & YANG, X. 2020. Urban big data fusion based on deep learning: An overview. *Information Fusion,* 53**,** 123-133.

MÜLLER, K.-R., SMOLA, A. J., RÄTSCH, G., SCHÖLKOPF, B., KOHLMORGEN, J. & VAPNIK, V. Predicting time series with support vector machines. International Conference on Artificial Neural Networks, 1997. Springer, 999-1004.

OKUTANI, I. & STEPHANEDES, Y. J. 1984. Dynamic prediction of traffic volume through Kalman filtering theory. *Transportation Research Part B: Methodological,* 18**,** 1-11.

SCHMIDT, R. M. 2019. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research,* 15**,** 1929-1958.

SUTSKEVER, I., VINYALS, O. & LE, Q. V. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

TEDJOPURNOMO, D. A., BAO, Z., ZHENG, B., CHOUDHURY, F. & QIN, A. 2020. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*.

VAN LINT, J. & VAN HINSBERGEN, C. 2012. Short-term traffic and travel time prediction models. *Artificial Intelligence Applications to Critical Transportation Issues,* 22**,** 22-41.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. & POLOSUKHIN, I. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems 30 (Nips 2017),* 30.

VICROADS 2019. Transport Modelling Guidelines (Volume 4).

VLAHOGIANNI, E. I., GOLIAS, J. C. & KARLAFTIS, M. G. 2004. Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews,* 24**,** 533-557.

VLAHOGIANNI, E. I., KARLAFTIS, M. G. & GOLIAS, J. C. 2014. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies,* 43**,** 3-19.

WANG, X., AN, K., TANG, L. & CHEN, X. 2015. Short term prediction of freeway exiting volume based on SVM and KNN. *International Journal of Transportation Science and Technology,* 4**,** 337-352.

WANG, Z., SU, X. & DING, Z. 2020. Long-Term Traffic Prediction Based on LSTM Encoder-Decoder Architecture. *IEEE Transactions on Intelligent Transportation Systems***,** 1-11.

WILLIAMS, B. M. & HOEL, L. A. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering,* 129**,** 664-672.

YU, B., SONG, X., GUAN, F., YANG, Z. & YAO, B. 2016a. k-Nearest neighbor model for multiple-time-step prediction of short-term traffic condition. *Journal of Transportation Engineering,* 142**,** 04016018.

YU, H.-F., RAO, N. & DHILLON, I. S. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. NIPS, 2016b. 847-855.

ZHANG, J., WANG, F.-Y., WANG, K., LIN, W.-H., XU, X. & CHEN, C. 2011. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems,* 12**,** 1624-1639.

ZHANG, L., LIU, Q., YANG, W., WEI, N. & DONG, D. 2013. An improved k-nearest neighbor model for short-term traffic flow prediction. *Procedia-Social and Behavioral Sciences,* 96**,** 653-662.

ZHENG, C., FAN, X., WANG, C. & QI, J. Gman: A graph multi-attention network for traffic prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 2020. 1234-1241.

ZHOU, H., ZHANG, S., PENG, J., ZHANG, S., LI, J., XIONG, H. & ZHANG, W. 2020. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *arXiv preprint arXiv:2012.07436*.